7-2013

A Test for Detecting Changes in Closed Networks Based on the Number of Communications Between Nodes

Christopher S. Wichman
University of Nebraska-Lincoln, wichmanc@cox.net

A TEST FOR DETECTING CHANGES IN CLOSED NETWORKS BASED ON THE NUMBER OF COMMUNICATIONS BETWEEN NODES

by

Christopher S. Wichman

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Statistics

Under the Supervision of Professor David B. Marx

Lincoln, Nebraska

July, 2013

A TEST FOR DETECTING CHANGES IN CLOSED NETWORKS BASED ON THE NUMBER OF COMMUNICATIONS BETWEEN NODES

Christopher S. Wichman, Ph.D.

University of Nebraska, 2013

Adviser: David B. Marx

This dissertation presents a formal method for detecting changes in a closed communications network based on an "abnormal" shift in the number of communications between some of the nodes. The method relies on the analyst's ability to define the network of interest; capture the number of communications between nodes; and to establish a history of normal communications flow between nodes over fixed intervals of time. A metric multi-dimensional scaling technique is then used to represent the network at each time interval with a k-dimensional (k = 1, 2, …) configuration. The affine bi-dimensional regression coefficient of determination (aR^2) between all adjacent time periods is calculated and recorded. As time progresses, the configuration and aR^2 are found and compared to the historical aR^2's. When a time period's aR^2 is abnormally low relative to the history, a change in the number of communications has been detected.

A simulation study was conducted using a closed network made up of ten nodes and three different edge density values (low, moderate, and high) to randomly generate the edges (connections) between nodes. A Poisson AR(1) process was used to generate the number of communications between nodes at each time period. Changes were then

randomly assigned in time periods 26 and 52, and the aR^2's calculated between adjacent time periods. A separate simulation was conducted for each combination of edge density (3 levels), AR(1) correlation parameter (3 levels), number of edges perturbed (3 levels), perturbation factor (3 levels), time period of perturbation (2 levels), and configuration dimension (2 levels). The results suggest that under these conditions the method as proposed has reasonable power for detecting "abnormal" changes in the number of communications.

ACKNOWLEDGEMENTS

This dissertation could not have been written without the advising and mentoring of Dr. David Marx. He and the other committee members, Dr. Stephen Kachman, Dr. Ashok Samal, and Dr. Dong Wang, gave of their time and expertise to help guide me through my academic program. I thank Dr. Erin Blankenship for mentoring my development as a teacher of statistics, and Dr. Martin Frenzel for challenging me in the multiple classes we took together. I would also like to thank my parents for providing me with an excellent foundational education. Finally, I would like to express my thanks and gratitude to my wife, Tammy, and our five children, Brittany, Meredith, Joshua, MacKenzie, and Samantha for their patience, support, and confidence in me while I pursued my new career.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Suppose that one is interested in detecting a change in the level or quantity of communications between *n* entities whose functions and relationships are understood. But, for whatever reason (the volume of communications, privacy laws, sovereignty issues, etc.), you cannot have direct access to the communications themselves. For example, consider a deterrence scenario. Country Orange states they are going to invade country Red if certain concessions are not made. Orange begins a military build-up on the border with Red. Blue begins making overtures to both Red and Orange to try and prevent an invasion. Understanding the communications volume between entities within Red and Orange prior to the tension between Red and Orange, in the build-up phase of tensions, and after Blue's overtures, can give insight into when and if the invasion will occur. For example, during the build-up phase, the governing body of Orange may communicate approximately evenly between military and diplomatic entities. After Blue's contact, a shift in communication towards the diplomatic arm might indicate that Blue's actions are yielding the desired de-escalation effect; a shift in communications volume towards the military may indicate a shift towards military action.

In the deterrence example, the inter-working of the Red, Orange, and Blue government organizations can be viewed as a network. Social scientists define a network as a set of entities, together with a social relation on those entities. "The social network

field is an interdisciplinary research program which seeks to predict the structure of relationships among social entities, as well as the impact of said structure on other social phenomena. The substantive elements of this program are built around a shared core of concepts and methods for the measurement, representation, and analysis of social structure". The quote taken from Butts' 2008 paper *Social Network Analysis: a methodological introduction* shows how the current state of network analysis is primarily focused on characterizing and describing a given network, based on social relationships and behaviors.

For the purposes of this work, the social science definition is adapted; removing the emphasis on social behavior and "relationships" to focus on the lines of communication. The Merriam-Webster® dictionary has five definitions of network; the third and fourth of which are most applicable to the work presented here: a) an interconnected or interrelated chain, group, or system; b) a system of computers, peripherals, terminals, and databases connected by communications lines. Merging these two definitions and focusing on the entity to entity[1] communication link, a network is an interconnected group where the entities making up the group are connected via communications lines.

For the example given, each department within the government organization represents an entity; the function (purpose) of each department defines the relationship each department has with the others, and the first communication between departments represents a link. Now, suppose that the underlying structure of the network has not changed, but the volume or flow of communication has changed. The goal of this work is

[1] An entity can be an individual person, a collection of people, or an organization.

to be able to detect a change in a network which is measureable based purely on the number of communications between nodes (the individuals or entities) in the network. Here, the primary measurement will be the number of communications between entities in a given time frame.

1.1 Definitions and Terminology

In general, there are two ways to represent network data; graph theory notation and matrix representation. The graph theory notation is used to define network measurements and to explain concepts regarding network analysis and summary. One class of measurements is called structural indices which are defined on one of two levels, node level and graph level [Butts 2008]. The matrix representation, allows the researcher to visualize the network in a numerical and pictorial way that is easy to understand. The entities or individuals making up the network are referred to as vertices or nodes in the graph theory or matrix notations, respectively. For the remainder of this work, the term node will be used exclusively when discussing both methods of representing a network.

In graph theory notation, a network is considered a graph, where each graph describes the relation between a set of nodes and a set of edges, i.e. $G = (V, E)$ [Butts, 2008]. Here the term edge refers to a link between nodes. Using Butts' (2008) notation, $V(G)$ represents the set of nodes and $E(G)$ the set of edges for a given graph G. Here the elements of $V(G)$ are represented by v, and the elements of $E(G)$ are represented by e. Therefore the size, $n = |V(G)|$, of the network is equal to the total number of nodes in the network.

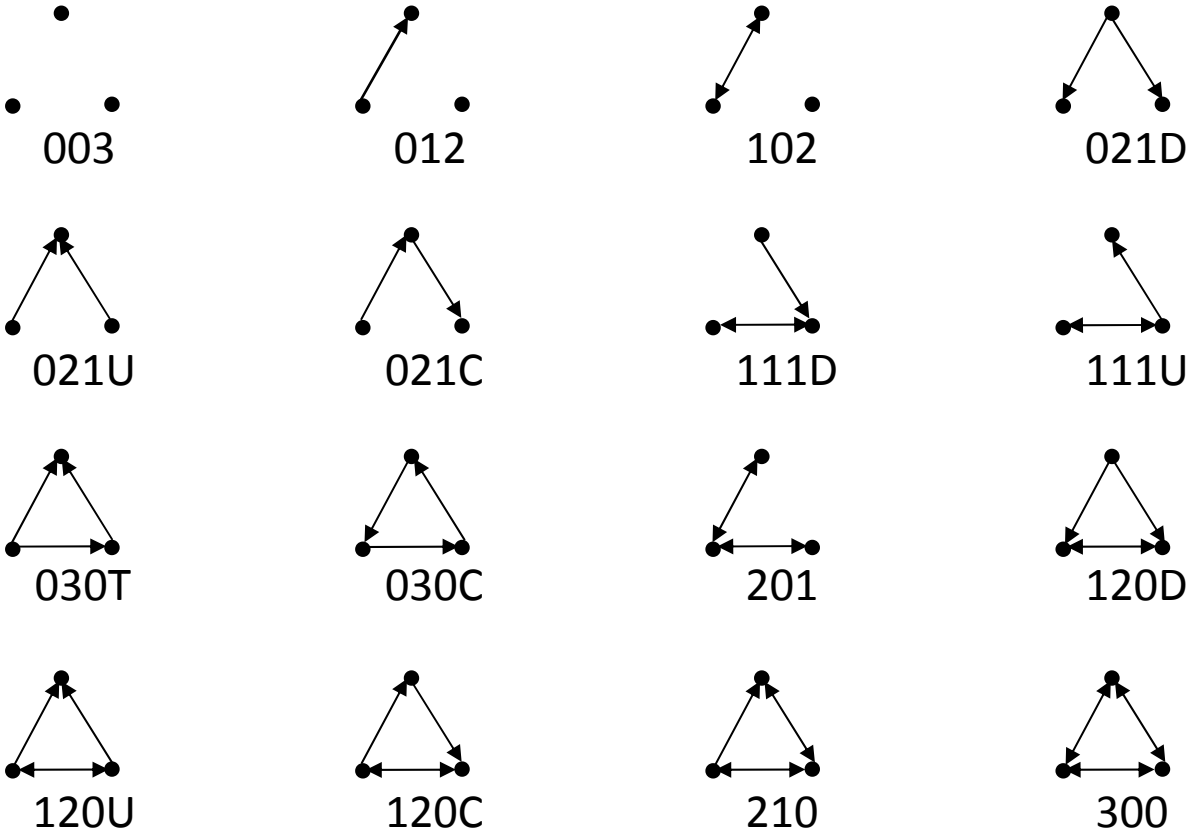
1.1.1 Graph Level Indices

Graph level indices (GLI) are measures of network structure which takes into account the entire network. A census of sub-graphs contained in the network graph is an example of a GLI. A sub-graph is a subset of the whole graph taken *k* distinct nodes at a time. The fundamental sub-graph in social network analysis is the pair-wise ($k = 2$) graph [Butts, 2008]. There are two classes of pair-wise graphs; directed pairs and undirected pairs. A directed pair is called a digraph [Holland & Leinhardt, 1976]. Digraph stands for directed graph and is defined as a set of ordered pairs of nodes and their associated edge set. In a digraph, the link from node *v* to *v'* may exist, but the link between *v'* to *v* may not. In undirected pairs, the direction of the link is not taken into account. Pair-wise graphs give rise to the first network structural element to be discussed, the dyad census.

A pair of any two distinct nodes is called a dyad. Dyads are assigned a three digit code based on the character of the dyad. Dyads can be either, mutual (M), asymmetric (A) or null (N) in character, giving rise to the MAN notation [Davis and Leinhardt, 1972]. A mutual dyad means that both nodes have communicated with each other; an asymmetric dyad means that one node has communicated with the other node, but the communication was not reciprocated. A null dyad means there is no communication between the two nodes. The number of dyads, D, in a network is $D = \binom{n}{2}$. The dyad census is literally the tally of the number of mutual, asymmetric, and null dyads contained in a network. It should be noted that the dyad census for an undirected network will be made up of only mutual and null dyads.

The second network structural element is the triad. The triad refers to the communication links between any three nodes; it is readily apparent that dyads are necessarily nested within triads. Therefore, triads are classified by the number of each type of dyad found in the triad. For a directed network, there are 16 isomorphism classes (Davis and Leinhardt, 1972), see figure 1-1. For the purposes of this work, all dyads will either be null or mutual; asymmetric dyads are not considered, since the measure of importance is total volume of communication between nodes and not the direction of the communication. In the case of undirected networks, only the isomorphisms, 003, 102, 201, and 300 are applicable.

Figure 1-1: Sixteen isomorphism classes with MAN notation [from Faust, 2006]



1.1.2 The Adjacency Matrix

As previously stated, the line of communication between any two nodes is termed an edge. An edge takes on one of two values: zero or one. Zero, if there is no communication between the two nodes. One, if there is any communication between the two nodes. The network can be represented as an adjacency matrix of ones and zeroes. Two nodes are considered adjacent if there is a non-zero edge between them. A network containing n nodes is said to have size, n. The adjacency matrix, $\boldsymbol{X}$, will have dimension n x n, where the i^{th} row and column elements represent the i^{th} nodes communications links with the other $(n-1)$ nodes. There are two types of adjacency matrices depending on the research question being investigated. First, if the only concern is whether or not there has been contact between two nodes, the adjacency matrix is symmetric (undirected). If nodes i and j have communicated during the time frame of observation, then

$$X_{ij} = X_{ji} = \begin{cases} 1 & \text{if nodes } i \text{ and } j \text{ have communicated} \\ 0 & \text{otherwise} \end{cases}$$

Second, if the investigation requires taking into account the direction of the communication, then the adjacency matrix is referred to as asymmetric (directed) and is split about the main diagonal where:

$$X_{ij} = \begin{cases} 1 \text{ if } i \text{ has commnuicated with } j \\ 0 \text{ otherwise} \end{cases} ; and\ X_{ji} = \begin{cases} 1 \text{ if } j \text{ has commnuicated with } i \\ 0 \text{ otherwise} \end{cases}$$

In an asymmetric adjacency matrix, the upper triangle takes into account the outward communication to each node and the lower triangle takes into account the inward

communication from each node. For the purposes of this work, self-communication is ignored, and therefore the main diagonal will be all zeroes.

1.1.3 Node Level Indices

Node level indices (NLI) focus on summarizing the characteristics of a given node in the network. There are numerous NLIs defined by social scientists, for a lengthy discussion see Butts, 2008 or Holland and Leinhardt, 1976. In this section, only the NLI, degree, will be discussed. The degree of a node is its number of adjacent connections. In an undirected network; the degree of a node can be found by taking the row sum (X_{i+}), or equivalently column sum $\left(X_{+j}\right)$ of the adjacency matrix. In graph theory notation, the degree of a node in an undirected network is $c_d(v, G) = |N(v)|$ [Butts, 2008]. If the network is directed, then the degree of the node has two measurements, the in-degree $c_{d+}(v, G) = |N^+(v)|$ and the out-degree $c_{d-}(v, G) = |N^-(v)|$. The row sum of the adjacency matrix for a given node yields the out-degree, and the column sum yields the in-degree.

1.1.4 Software Packages for Analyzing Networks

There are two major software packages dedicated to the exploration, summary, analysis, and simulation of social networks, UCINET (Borgatti, et al., 1992) and STATNET (Handcock, et al., 2008). Both packages offer the user a graphing / plotting capability that allows the user to visualize the node to node connections in the network (section 1.2); summary measurements that describe the structure of the network, as well as the ability for the user to simulate the growth of a network (not applicable to the work done here). Of the two packages, UCINET is the more user friendly for computer

programming illiterates, as it is windows based and the different analyses are each accessible from drop down menus in a point and click format. STATNET is the more powerful of the two packages when one is looking to simulate the growth of networks as it incorporates the latest exponential random graph modeling techniques; however, the package is based in the R statistical language and requires the user to be familiar with programming techniques to unlock its full potential.

Both packages provide the user with functions that will calculate common measures used in the study of social networks; these include: the **size** of the network (n) also known as the total number of nodes (egos) in the network; the number of **edges** (aka ties) in the network; the number of possible **distinct nodal pairs** (i.e. n choose 2); **density** (number of edges/pairs); **average geodesic distance** (average shortest path length between nodes); **Diameter:** the maximum geodesic distance in the network; **two step reach** (i.e. the number of nodes within a two step reach of the target node; **reach efficiency** (2 step reach / n); and **un-reach** (the number of pairs with an infinite distance (no connections). In addition to the aforementioned statistics, STATNET and SNA provide the user with the total number of each of the 16 possible triad configurations and the number of each of the three dyad configurations, as well as other statistics that will be defined as needed throughout this work.

1.2 Motivation

The current state of network analysis is to predict the direction and speed of growth of open networks, as well as to model why certain relationships exist and others do not. Network analysts have developed formal methods for detecting these structural changes [Butts, 2008]. However, the analysis of the number of communications or some other edge attribute is ad-hoc. Typically, the attribute is represented by weighting the edge line in the network plot relative to the size of the attribute [NETDRAW®, STATNET]. If an edge line gets thicker between time periods of measure, then a change is deemed to have occurred. This method necessarily requires the grouping of attribute size into bins. A limitation is that a single communication can tip the scale causing the edge line to change, and thus a change in the network will have "occurred". The goal of this dissertation is to develop a more formal method for detecting changes in a network based solely on the number of communications between nodes.

Any method for detecting such a change needs to account for the degree of each node, since an overall increase in communications along all edges linked to a single node is different than the same size increase occurring at random throughout the network. In other words, we are trying to keep the network structure intact. Network analysts have devised numerous methods for visualizing networks with an emphasis on producing aesthetically pleasing plots. Each of these methods and the reason they were not chosen for this work are explained in the next section. A method which does capture the number of communications along each edge, while preserving the degree information is multi-dimensional scaling. There are multiple types of multi-dimensional scaling; the two most

promising for this work are discussed at length in chapter 2. The reduction in dimension provided by multi-dimensional scaling (the configuration) lends itself to comparison via configuration matching techniques. Four configuration matching techniques were considered and each is discussed in chapter 2. Of the four, affine bi-dimensional (Tobler, 1994), tri-dimensional (Schmid, et al, 2012) regression were found to provide the most promise for this work. Since there is the possibility that the number of communications between nodes cannot adequately be represented in either two or three dimensions, part of this dissertation extends bi- and tri-dimensional regression to k-dimensional regression ($k > 3$).

1.3 Visualizing Networks

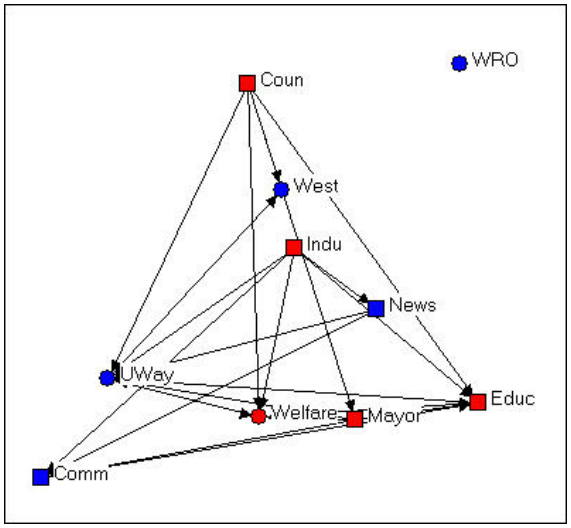
1.3.1 Plotting Capabilities: NETDRAW ®

A picture, graph or drawing is often desired so that an analyst can "see" what a network looks like. In this regard, both network analysis software packages do an excellent job of representing networks as a "snapshot in time". NETDRAW ® (the plotting package utilized by UCINET®) provides the user with multiple options for displaying the connections within a network including directed and undirected ties. The Layout options are (in the order presented by Hanneman et al in the e-text "Introduction to Social Network Methods"): 1) Random, 2) Circle, 3) Multivariate Scaling, and 4) Spring Embedding.

The layout option *random* [Figure 1-2] plots the individual nodes in a random directed fashion. Essentially, the first node to be plotted and its coordinate is chosen at

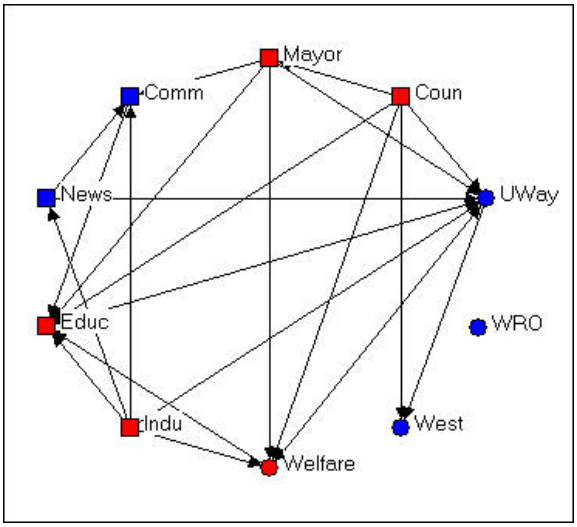
random and placed on the plot, a second node is selected at random and placed in the plot, if these nodes have a connection a line is drawn between the two. If the tie is directed, then the line is drawn as an arrow showing the direction of the tie. This process continues until every node and its connections are represented in the drawing.

Figure 1-2: Example of the layout option *random* from NETDRAW®.



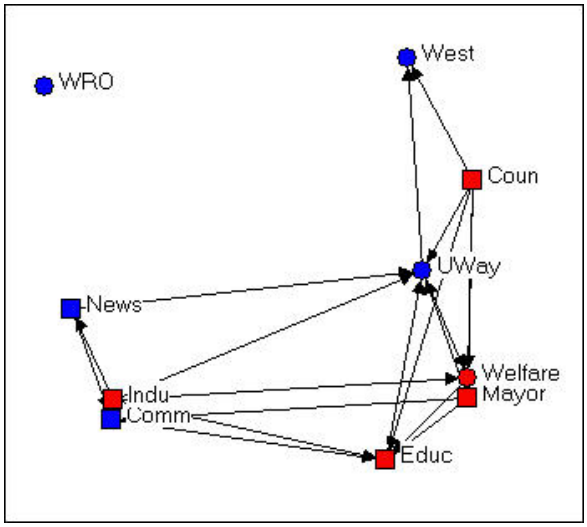
The layout option *circle* [Figure 1-3] also plots the individual nodes in a random directed fashion, only this time the node coordinates are randomly selected to lie on the circumference of a circle of fixed radius. Essentially, the first node to be plotted and its coordinate is chosen at random and placed on the circle, a second node is selected at random and placed on the circle, if these nodes have a connection a line is drawn between the two. If the tie is directed, then the line is drawn as an arrow showing the direction of the tie. This process continues until every node and its connections are represented in the drawing.

Figure 1-3: Example of the layout option *circle* from NETDRAW®.



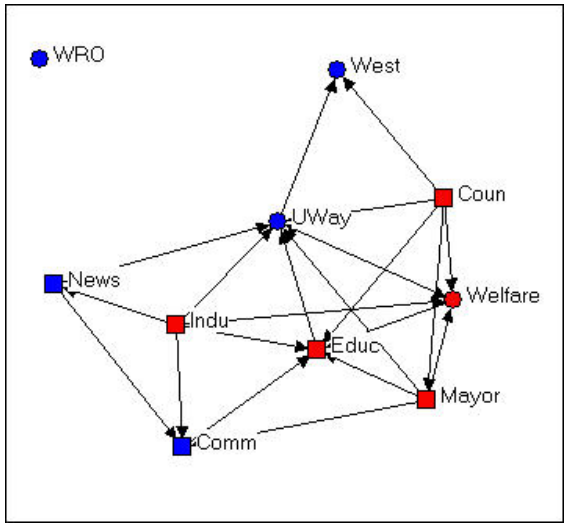
The layout option *MDS* [Figure 1-4], uses the multivariate technique of multidimensional scaling to generate the coordinates of each node. NETDRAW ® provides the user the ability to select which MDS algorithm (metric or non-metric) to use in determining the coordinates. Various MDS algorithms will be discussed in detail in chapter 2. In NETDRAW ®, the geodesic distance of each node is used as a measure of similarity (or dissimilarity) for the purposes of the MDS algorithm chosen. This results in nodes with similar geodesic distances being placed closer together on the plot than those with dissimilar geodesic distances. For example three nodes with geodesic distances of 3, 3, 4 will be closer together in the plot, than three nodes with geodesic distances of 3, 7, 1.

Figure 1-4: Example of the layout option *MDS* from NETDRAW®.



The last layout option, *spring embedding* [Figure 1-5], uses a heuristic algorithm developed by Eades (1984) to provide a clearer picture of networks based on the concept of the minimization of mechanical energy (or "near" spring force equilibrium). A heuristic is a method which arrives at a solution through trial and error versus a formal set of principles.

Figure 1-5: Example of the layout option *spring embedding* from NETDRAW®.



In Eades' algorithm, each node is treated as a metal ring (i.e. rigid and does not distort due to spring force) connected to the nodes with which it has a tie by a spring with known spring constant. Next, imaginary springs are placed between nodes which have ties, and each spring is provided a spring constant. The system is "let go" and allowed to come to rest within fixed, circular boundaries. This process is repeated until the mechanical system energy is a minimum (sum of spring force is at equilibrium). A more detailed explanation follows.

Eades' goal was to develop a "program which assigns locations to vertices in such a way that the resulting layout is in some sense aesthetically pleasing." This goal gave rise to two criteria: 1) edge lengths for connected nodes should be approximately the same, and 2) the plot layout should be symmetric. Eades first randomly assigns nodes a location in two-dimensional space (if nodes end up on top of each other, one is randomly moved until all nodes are "unmasked"). Next, he treats the network as a virtual mechanical system in which each node is treated as a rigid, metal ring and each edge is treated as a spring with known spring constant (Force/meter) and natural length (length of the un-stretched /un-compressed spring), d_0 . Further, to reduce the number of calculations required and to accommodate the fact that Hooke's Law [see equation 1-1] is only valid over a very small deviation from a spring's natural length, Eades only looks at the effect of the spring force between adjacent nodes, and treats all non-adjacent nodes as repulsive.

$$F(Spring) = -k(x - x_0) \quad (1.1)$$

Where: x is the distance the spring is stretched / compressed from x_0, and x_0 is the spring's natural length.

Arguing that "Experience shows that Hooke's Law (linear) springs are too strong when the vertices are far apart; the logarithmic force solves this problem." Eades re-defines the spring force due to adjacent nodes to be logarithmic:

$$F_{Spring} = C_1 \log\left[{d_0}/{C_2}\right] \quad (1.2)$$

Where C_1 and C_2 are constants, and d_0 is the spring length (either stretched or un-stretched). It is important to note that by adopting the logarithmic force, Eades is requiring the force between adjacent nodes to be either attractive or zero. There is no possibility of a spring repulsive force, because when d_0 is negative (i.e. the spring is repelling the two nodes) equation 1-2 is not defined.

For nonadjacent nodes, Eades drops the spring effect entirely and treats the nodal interaction as repulsive. Eades likens the repulsive force for nonadjacent nodes to a Coulomb force law (i.e. the repulsive force is proportional to the inverse square of distance between nodes) defined as:

$$F_{i\ repels\ j} = {C_3}/{d_{ij}^2} \quad (1.3)$$

Where C_3 is a constant and d_{ij} is the distance between node i and node j.

With the forces defined, the algorithm works as follows:

1. Randomly select one of the *n* nodes and generate a coordinate, plot the node (Repeat *n* times) [2]
2. Calculate the net force on each of the n nodes, due to the other (n – 1) nodes. Allow each node to move $\Delta_i = C_4 * net\ force\ on\ node\ i$ in the direction of its net force (see figure 1-6).
3. Repeat step 2. M times or until $\max(\Delta_i) < threshold\ value$.
4. Plot the final graph.

Eades proposes that the values 2, 1, 1, 0.1 be used for C_1, C_2, C_3, and C_4, respectively.

Example: A single iteration of Eades' Spring Embedder.

Consider the following adjacency matrix for a four node network:

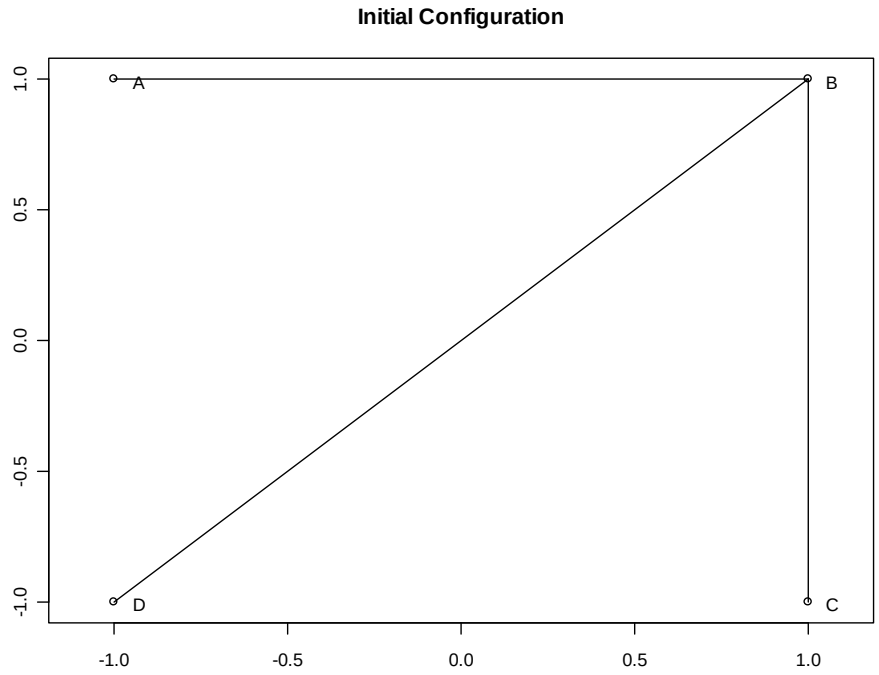
$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

And suppose the initial configuration is as follows: node A at (-1, 1); node B at (1, 1); node C at (1, -1) ; and node D at (-1, -1) , see figure 1-6. Before calculating the net force on each of the four nodes in the network, assumptions regarding the sign (positive or negative) of the force relative to the cardinal directions must be made. In this example, forces which act upward in the y direction and to the right in the x direction are considered to be the positive. Additionally, each force will be represented by the

[2] In practice, such as UCINET®, the spring embedded algorithm starts with an initial coordinate position based on an MDS solution.

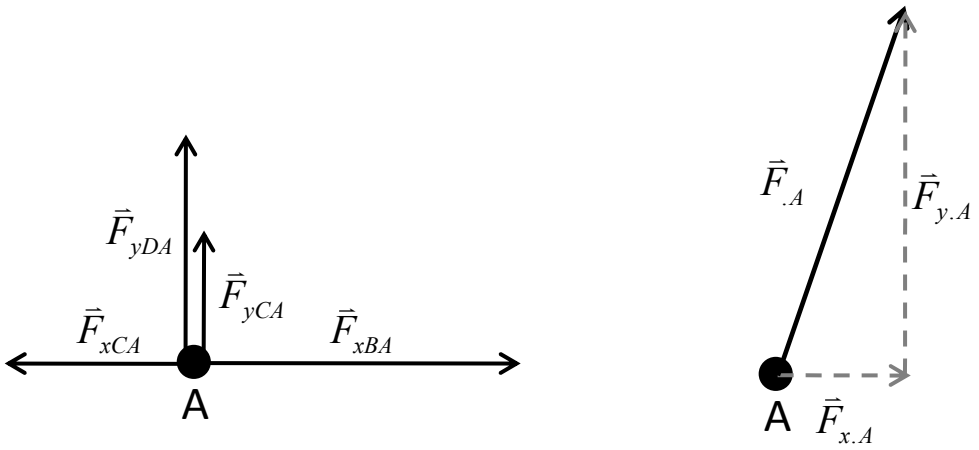
symbol, $\vec{F}_{iJK}$. Here $\vec{F}_{iJK}$ is read "the force due to node *J* acting on node *K* in cardinal direction *i*.

Figure 1-6: Four node network initial configuration.



From the adjacency matrix and the initial configuration in Figure 1-6, it is apparent that there is an attractive force due to node B on node A; and the forces on node A due to nodes C and D are repulsive. The component force resolution diagram and the resultant force diagram for node A can be found in Figure 1-7.

Figure 1-7: The force resolution diagram for node **A.**



The net force on node A in each of the cardinal directions is:

$$\vec{F}_{x.A} = \vec{F}_{xBA} - \vec{F}_{xCA}$$

$$\vec{F}_{y.A} = \vec{F}_{yBA} - \vec{F}_{xCA}$$

$$\vec{F}_{.A} = \sqrt{\left(\vec{F}_{x.A}\right)^2 + \left(\vec{F}_{y.A}\right)^2}$$

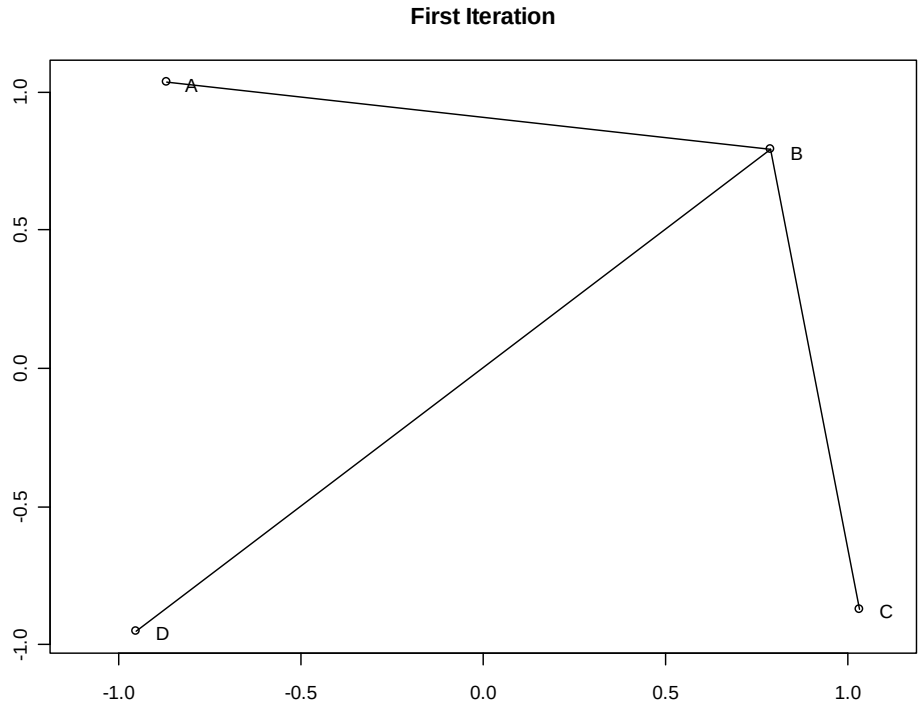
From the right hand picture in Figure 1-7, node A will move up and to the right from its initial position. The forces for nodes B, C, and D were found in a similar fashion (Table 1-1). As a result, after the first iteration, node B moves down and to the left of its initial position; and nodes C and D both move up and to the right of their initial positions (see Figure 1-8).

Applying Eades' step four to each node: node A will move to the right 0.13 units and up 0.034 units; node B will move left 0.21 units and down 0.21 units; node C will move to the right 0.034 units and up 0.13 units; and finally, node D will move to the right 0.05 units and up 0.05 units (Figure 1-8).

Table 1-1: Calculation of forces for nodes A, B, C, and D

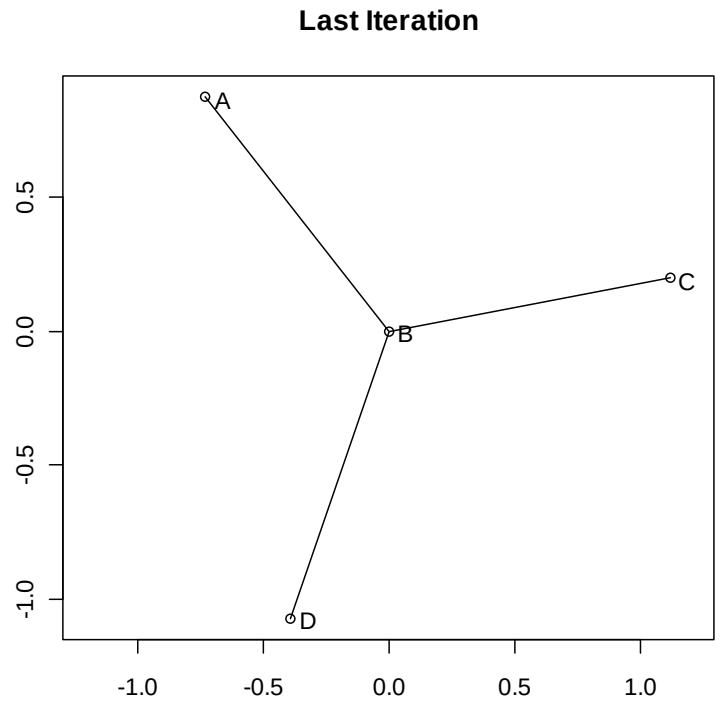
Node A:	$\vec{F}_{xBA} = 2\ln(2)$ $\vec{F}_{yBA} = 0$ $\vec{F}_{xCA} = -\cos(45^{\circ})\left(\frac{1}{8}\right)$ $\vec{F}_{yCA} = \cos(45^{\circ})\left(\frac{1}{8}\right)$ $\vec{F}_{xDA} = 0$ $\vec{F}_{yDA} = \frac{1}{4}$ $\vec{F}_{x.A} = 1.3$ $\vec{F}_{y.A} = 0.34$	Node B:	$\vec{F}_{xAB} = -2\ln(2)$ $\vec{F}_{yAB} = 0$ $\vec{F}_{xCB} = 0$ $\vec{F}_{yCB} = -2\ln(2)$ $\vec{F}_{xDB} = -\cos(45^{\circ})2\ln(\sqrt{8})$ $\vec{F}_{yDB} = -\cos(45^{\circ})2\ln(\sqrt{8})$ $\vec{F}_{x.B} = -2.12$ $\vec{F}_{y.B} = -2.12$
Node C:	$\vec{F}_{xAC} = \cos(45^{\circ})\left(\frac{1}{8}\right)$ $\vec{F}_{yAC} = -\cos(45^{\circ})\left(\frac{1}{8}\right)$ $\vec{F}_{xBC} = 0$ $\vec{F}_{yBC} = 2\ln(2)$ $\vec{F}_{xDC} = \frac{1}{4}$ $\vec{F}_{yDC} = 0$ $\vec{F}_{x.C} = 0.34$ $\vec{F}_{y.C} = 1.3$	Node D:	$\vec{F}_{xAD} = 0$ $\vec{F}_{yAD} = \frac{1}{4}$ $\vec{F}_{xBD} = \cos(45^{\circ})2\ln(\sqrt{8})$ $\vec{F}_{yBD} = \cos(45^{\circ})2\ln(\sqrt{8})$ $\vec{F}_{xCD} = -\frac{1}{4}$ $\vec{F}_{yCD} = 0$ $\vec{F}_{x.D} = 0.49$ $\vec{F}_{y.D} = 0.49$

Figure 1-8: Positions of nodes A, B, C and D after one iteration of Eades Spring Embedder Algorithm.



Notice in Figure 1-8, that relative to node B, node A has moved clockwise, and nodes C and D have moved counter-clockwise; and node B has moved toward the center relative to nodes A, C and D. Repeating Eades' algorithm approximately 100 times (dependent on the initial configuration and the choice of C4), the final configuration ends with node B in the center equidistant from nodes A, C and D; and Nodes A, C and D equidistant from each other (Figure 1- 9).

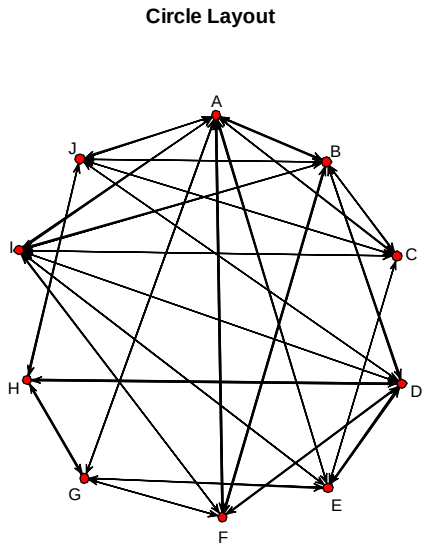
Figure 1-9: Positions of nodes A, B, C and D after 100 iterations of Eades Spring Embedder Algorithm.



1.3.2 Plotting Capabilities: plot.net:

Since statnet runs in an R environment, statnet and its associated plotting package are object oriented. As such, any adjacency matrix (or matrix of weights) that is tagged as a network object will automatically be plotted as a network plot using the simple command plot in R. There are three network layout options that can be set by the user; circle, Fruchterman-Reingold, or Kamada-Kawai. The Circle layout option in plot.network works the same as in NETDRAW ® and a sample output can be found in figure 1-10.

Figure 1-10: The *circle* layout option in plot.network



.

The Fruchterman-Reingold layout option uses the algorithm for graph drawing by forced directed placement from the 1991 work of Thomas Fruchterman and Edward Reingold. Fruchterman and Reingold (F and R) state that their goal was to create an algorithm which produced "aesthetically-pleasing, two-dimensional pictures of graphs by doing simplified simulations of physical systems." They started their work with two guiding principles: "1. Vertices connected by an edge should be drawn near each other.", and "2. Vertices should not be drawn *too* close to each other." The algorithm developed is similar to the work of Eades in that the overall system energy is minimized; however, Fruchterman and Reingold treat the problem as if each node is an "atomic particle or celestial body". When looking at nuclear forces there are some which are very weak and only exert influence over very small distances (i.e van Der waals force) and some which are strong, exerting force over extended ranges (i.e. Coulomb's forces) . Fruchterman

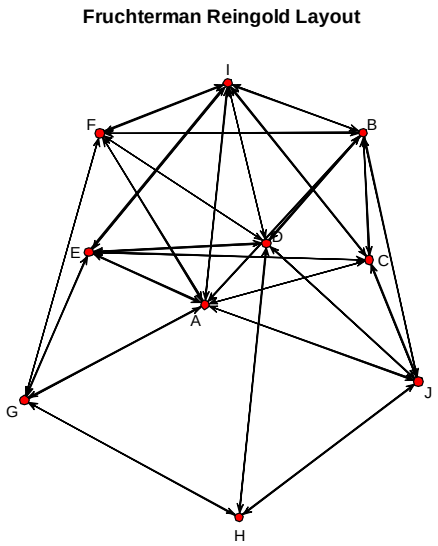
and Reingold run with this concept and modify physical reality by treating all attractive forces as weak (attractive force only operates on directly adjacent nodes; however in reality Coulomb's force is strong and can be both attractive and repulsive). They treat all repulsive forces as strong, and therefore all nodes which are connected can "feel" some amount of repulsion from other nodes.

The algorithm is iterative by node. To set the scene, picture a solid crystalline material at a temperature above absolute zero. When a solid is above absolute zero, there are necessarily inclusions and "cracks" in the material if the material is to be at its lowest energy state, thus allowing some atoms to have ties while others do not. For the purpose of the algorithm, treat each node as an atom of the crystal. In Crystal Theory, each atom in a crystalline structure is considered bound to its lattice location with the freedom to move back and forth in three dimensional space by a fixed amount determined by the temperature (a measure of internal energy) of the material. F and R only considered two-dimensional space and thus each node is treated as being fixed to a location, but able to freely move in a circle of fixed radius centered on the lattice location. The initial lattice structure (initial position for the nodes in the plot) is chosen at random.

Armed with the initial positions of each node, the algorithm progresses iteratively from node to node until a threshold, total system energy is reached. The algorithm has three steps for each node; the forces are resolved in the following order: attractive, then repulsive, and finally maximum displacement based on temperature. In other words, if the net force, attractive plus repulsive only moves the node a small distance that is within the maximum distance allowed by the materials temperature, the node is considered at its lowest energy state, and will not move again, unless the perturbation of another node

further along in the algorithm upsets the balance of forces. The resulting pictorial output [see Figure 1-11] is a network in which the relationships (ties) tends to be "clearer" to the observer than using the circle layout option, especially when dealing with networks with a large number of nodes.

Figure 1-11: *Fruchterman Reingold* layout from plot.network.



Specifically, Fruchterman and Reingold define the attractive force as

$$f_a(d_{ij}) = {d_{ij}^2}/{k} \quad (1.4)$$

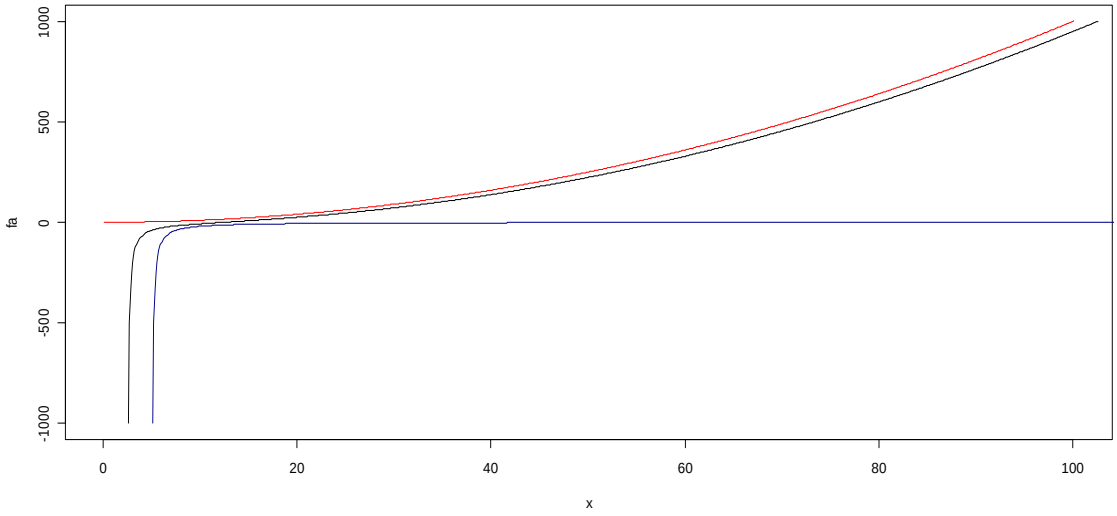
and the repulsive force as

$$f_r(d_{ij}) = -{k^2}/{d_{ij}} \quad (1.5)$$

Where $k = C\sqrt{{area\ of\ plot}/{number\ of\ nodes}}$ (1.6)

and C is a constant whose value is determined experimentally. The area of the plot is typically proportional to n^2; and k is defined as the optimal distance between nodes. The behavior of the forces is presented graphically in figure 1-12. From figure 1-12, it is apparent that the repulsive force is very strong, and overwhelms the attractive forces if any two nodes get too close together. Also, the repulsive force rapidly becomes negligible as the distance between nodes increases. The distance at which the repulsive force becomes negligible is dependent on the value of k, which in turn depends on the choice of C and the number of nodes in the network of interest.

Figure 1-12: Fruchterman and Reingold force versus distance. Here the attractive force is in red, repulsive force in blue, and the sum of forces in black.



The algorithm works as follows:

1. Randomly select one of the *n* nodes and generate a coordinate, plot the node (Repeat *n* times). Set the systems initial temperature to T_0.
2. Determine C and calculate *k* using equation 6.
3. Do for each node 1 to *n*:

a. Calculate the attractive force using equation 1.4.

b. Calculate the repulsive force using equation 1.5.

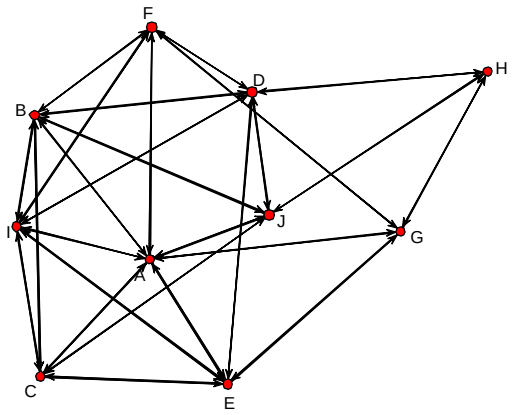
c. Calculate the sum of forces

4. Check current system temperature (temperature controls the maximum movement of the nodes when the system is "released". Release the system and allow it to stabilize.

5. Cool the system by decreasing the system temperature by a fixed amount (here, the algorithm is making the circle in which each node can travel smaller.

6. Repeat steps 3, 4 and 5 until the maximum displacement of the nodes is less than a researcher defined threshold.

Figure 1-13: Illustration of the effect of the random starting configuration on node placement in the *Fruchterman Reingold* option of plot.network.



An illustration of how the random starting lattice affects the drawing of the graph can be seen by comparing figures 1-11 and 1-13. This demonstrates, pictorially, that the algorithm finds a local minimum for total system energy, viz-a-viz a global minimum. This characteristic of the algorithm is the key reason the Fruchterman Reingold layout was not chosen for use in this work.

The final layout option in plot.network is Kamada-Kawai. In their 1989 work, Kamada and Kawai, presented an adaptation of Eades, 1984, spring embedder. In essence, they modified the Eades spring embedder by applying the requirement of a "desirable length" between nodes. In other words, they want each connected node in the picture to be separated by the same distance. Again, each node is treated as a metal ring connected by a spring to other nodes in the network, if two nodes do not have a connection, then no spring is present (the same as modeling a spring with a spring constant of 0).

The energy equation that Kamada and Kawai define for minimization is:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+j}^{n} \frac{1}{2} k_{ij} \left(|p_i - p_j| - l_{ij} \right)^2 \quad (1.7)$$

Where: l_{ij} (see equation 1.8) is the desirable distance between p_i and p_j.

$$l_{ij} = L \, X d_{ij} \quad (1.8)$$

$$L = {}^{L_0}\Big/_{\max\limits_{i<j} d_{ij}} \quad (1.9)$$

Here L_0 is the length of one side of the square display area. L is defined as the desirable length of a single edge in the display plane.

n is the number of nodes; p_i represents the position of node i; and k_{ij} (equation 1.10) is the spring "strength" for the spring connecting node i to node j.

$$k_{ij} = {K}/{d_{ij}^2} \quad (1.10)$$

Here K is an overall spring constant (ie in SI units Newtons per meter) and d_{ij} is the actual distance between two nodes at the start of the iteration.

Throughout the derivation of their algorithm it is assumed the network is symmetric, i.e. $k_{ij} = k_{ji}$ and $l_{ij} = l_{ji}$ $(i \neq j)$.

To minimize the system energy, equation 1.7 must first be rewritten into an accepted coordinate system form. In this case Kamada and Kawai chose the Cartesian coordinate system:

Expanding Equation 1.7

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left(|p_i - p_j|^2 - 2 l_{ij} |p_i - p_j| + l_{ij}^2 \right)$$

$$|p_i - p_j| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Where the ordered pair (x_i, y_i) represents the coordinates of node p_i

Substituting yields:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left((x_i - x_j)^2 + (y_i - y_j)^2 + l_{ij}^2 - 2 l_{ij} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right) \quad (1.11)$$

From calculus the condition that must be satisfied for there to be a local minimum is:

$$\frac{\partial E}{\partial x_m} = \frac{\partial E}{\partial y_m} = 0 \; for \; 1 \leq m \leq n \qquad (1.12)$$

Differentiating equation 1.11 with respect to x and y yields equations 1.13 and 1.14, respectively.

$$\frac{\partial E}{\partial x_m} = \sum_{j \neq m} k_{mj} \left\{ (x_m - x_j) - l_{mj}(x_m - x_j) \Big/ \sqrt{(x_m - x_j)^2 + (y_m - y_j)^2} \right\} \qquad (1.13)$$

$$\frac{\partial E}{\partial y_m} = \sum_{j \neq m} k_{mj} \left\{ (y_m - y_j) - l_{mj}(y_m - y_j) \Big/ \sqrt{(x_m - x_j)^2 + (y_m - y_j)^2} \right\} \qquad (1.14)$$

There are a total of 2n of these equations which ideally would be solved simultaneously to find the local minimum; however, these equations are not independent due to the fact that in a spring system if one metal ring is allowed to move, the total spring energy changes, and the force due to the moving ring as felt by the nodes attached to it via a spring will necessarily change [see figure 1-7]. Thus Kamada and Kawai use an iterative process in which only one node is allowed to move per iteration, and utilize a Newton – Raphson method to minimize the system energy.

At time t = 0, the node with the largest Δ_m defined as

$$\Delta_m = \sqrt{\left\{\frac{\partial E}{\partial x_m}\right\}^2 + \left\{\frac{\partial E}{\partial y_m}\right\}^2} \qquad (1.15)$$

is selected for minimization in the current Newton-Raphson step. From the starting point $\left(x_m^{(t)}, y_m^{(t)}\right)$ a new position $\left(x_m^{(t+1)}, y_m^{(t+1)}\right)$ is found by iterating the following step:

$$x_m^{(t+1)} = x_m^{(t)} + \partial x;\ y_m^{(t+1)} = y_m^{(t)} + \partial y \quad for\ i = 1, 2, 3, \ldots \quad (1.16)$$

The ∂x and ∂y are found by simultaneously solving the following two equations:

$$\frac{\partial^2 E}{\partial x_m^2}\left(x_m^{(t)}, y_m^{(t)}\right)\partial x + \frac{\partial^2 E}{\partial x_m \partial y_m}\left(x_m^{(t)}, y_m^{(t)}\right)\partial y = -\frac{\partial E}{\partial x_m}\left(x_m^{(t)}, y_m^{(t)}\right) \quad (1.17)$$

$$\frac{\partial^2 E}{\partial y_m^2}\left(x_m^{(t)}, y_m^{(t)}\right)\partial y + \frac{\partial^2 E}{\partial y_m \partial x_m}\left(x_m^{(t)}, y_m^{(t)}\right)\partial x = -\frac{\partial E}{\partial y_m}\left(x_m^{(t)}, y_m^{(t)}\right) \quad (1.18)$$

The partial derivatives in equations 1.17 and 1.18 come from taking the appropriate partial of equations 1.13 and 1.14. It is worth noting here that $\frac{\partial^2 E}{\partial y_m \partial x_m}\left(x_m^{(t)}, y_m^{(t)}\right) = \frac{\partial^2 E}{\partial x_m \partial y_m}\left(x_m^{(t)}, y_m^{(t)}\right)$ and therefore this partial only needs to be taken once either starting with equation 1.13 or equation 1.14. Taking the aforementioned partial derivatives yields:

$$\frac{\partial^2 E}{\partial x_m^2}\left(x_m^{(t)}, y_m^{(t)}\right) = \sum_{j \neq m} k_{mj}\left\{1 - l_{mj}\left(y_m - y_j\right)^2 \Big/ \left(\left(x_m - x_j\right)^2 + \left(y_m - y_j\right)^2\right)^{1.5}\right\} \quad (1.19)$$

$$\frac{\partial^2 E}{\partial y_m^2}\left(x_m^{(t)}, y_m^{(t)}\right) = \sum_{j \neq m} k_{mj}\left\{1 - l_{mj}\left(x_m - x_j\right)^2 \Big/ \left(\left(x_m - x_j\right)^2 + \left(y_m - y_j\right)^2\right)^{1.5}\right\} \quad (1.20)$$

$$\frac{\partial^2 E}{\partial x_m^2}\left(x_m^{(t)}, y_m^{(t)}\right) = \sum_{j \neq m} k_{mj}\left\{\frac{l_{mj}(y_m - y_j)(x_m - x_j)}{\left((x_m - x_j)^2 + (y_m - y_j)^2\right)^{1.5}}\right\} \tag{1.21}$$

The algorithm will cease and output a plot when either the maximum number of iterations is reached or the max (Δ_m) falls below some preset threshold value.

Similar to Fruchterman and Reingold, Kamada and Kawai's optimization algorithm starts with a random configuration of points. This again leads to a local versus global minimum for total system spring energy. The pictures yielded from two separate runs of the algorithm yield different results, see figures 1-14 and 1-15.

Figure 1-14: *Kamada Kawai* layout in plot.network.

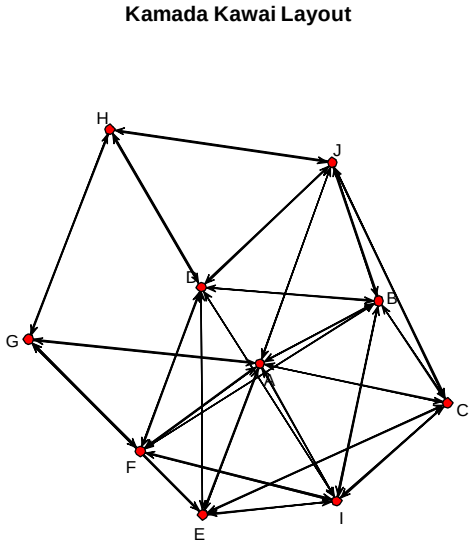
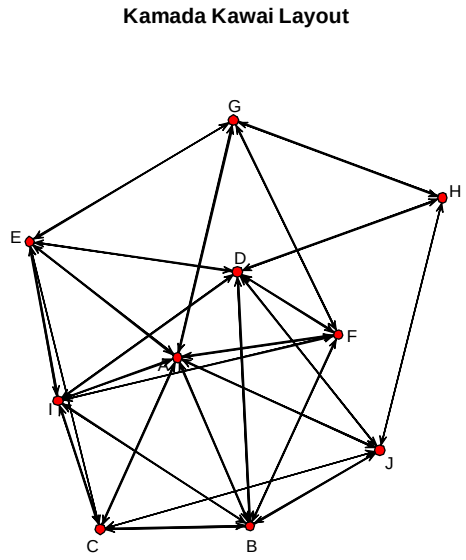


Figure 1-15: Second run of the *Kamada Kawai* layout option in plot.network.



Notice that in Figure 1-14, the node F shows up on the outer edge of the plot, where as in Figure 1-15, node F is in the interior of the plot. This characteristic of the algorithm is the key reason the Kamada and Kawai layout was not chosen for use in this work. There are other algorithms based on minimization of system energy currently in use where, in addition to striving for uniform length between nodes, the number of edge crossings is minimized. This additionalrequirement does nothing to change the overall unsuitability of the three layout options in plot.network; they are all essentially random processes.

1.4 Closed vs Open Networks

In order to identify the nodes that are important to observe during a network analysis, one must be able to bound the size of the network. From Butts (2008) "A network is bounded by the set of entities on which it is defined". Butts goes on to define three types of network boundaries: 1) the exogenously defined boundary; 2) the relationally defined boundary; and 3) the methodologically defined boundary.

The exogenously defined boundary is one in which the researcher "has a clearly specified substantive theory which indicates the entities (whether they be organizations or individuals) that are relevant for some phenomenon of interest" [Butts]. The relationally defined boundary is one in which the researcher knows the relationships of interest and that there is no interaction outside of the identified network which will influence the "in network" relationships. The methodologically defined boundary is one in which the sampling method to obtain the entities to be studied naturally limits the scope of the network. Here, Butts uses the example of sampling based only on the medium through which the relationship between entities is maintained (e.g. telephone, or e-mail, etc).

For the purposes of this work, networks will be broken into two broad categories. First, an open network is a network with no discernible boundaries; the entities that are interconnected are exclusively individuals. A prime example of an open network is the world-wide web. If a person were to trace every connection, the network would spread across multiple continents and contain millions if not billions of individuals. With sampling at each time point the number of people gaining access to the web may

fluctuate. A closed network on the other hand, is essentially a combination of the three network boundaries defined by Butts. It will be assumed that the researcher conducting the network study will have a prior well defined theory as to the entities of interest; the method of communication between entities is well defined; and that the relationship between entities is well understood.

1.5 Detecting Changes in Networks

When this work was originally started, the goal was to develop a statistic that could be used to test if a significant change had occurred in a network between two time periods. The statistic considered was degree variance. The focus of this first attempt was on undirected networks, and therefore the general definition of degree was used. Two problems arose; first, edges can flop between nodes and there are instances where the degree variance will not change, and therefore, the change in the network would not be detected. The second problem, arose when it was realized that the degree variance is actually a measure of structure versus a measure of spread in communication volume. Additionally, network analysts have already developed methods for detecting structural changes.

1.5.1 Detecting Structural Changes in Networks

Over the past two to three decades, work on detecting changes in a network has focused on detecting structural changes. Two of the methods will be discussed here. The first method is referred to as a linear subspace method [Butts, 2008], and is literally a measure of covariance between the adjacency matrices of the two networks. Essentially

one is finding the covariance between the two vectorized adjacency matrices with the diagonal removed. The common form of the graph covariance is defined to be:

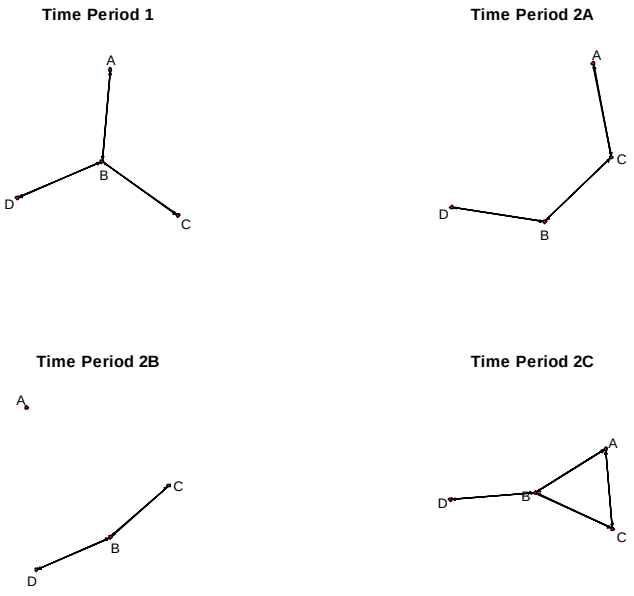
$$cov(G, G') = \frac{1}{(n^2 - n - 1)} \sum_{i=1}^{n} \sum_{j=1}^{n} (y_{ij} - \mu)(y'_{ij} - \mu'); i \neq j \quad (1.22)$$

Rewriting equation 1.22 in vector notation gives:

$$cov(G, G') = \frac{t(\boldsymbol{Y} - \mu\mathbf{1})(Y' - \mu'\mathbf{1})}{(n^2 - n - 1)} \quad (1.23)$$

Where *Y* and *Y'* are the adjacency matrices associated with *G* and *G'* respectively, and μ and μ' are the means of *Y* and *Y'*, respectively. Noting that the $var(G) = cov(G, G)$; the graph correlation $\rho(G, G') = cov(G, G')/\sqrt{var(G)var(G')}$.

Figure 1-16: Graph Covariance Examples.



Suppose you are looking to detect changes in a network over time, say between time period one and time period two. If there is no change in adjacency between the two time periods, then obviously the graph correlation will be 1. Notice, that if edges flip (time period 2A in figure 1-16), cease to exist (time period 2B in figure 1-16), or appear in the time period between the observation of *G* and *G'* (time period 2C in figure 1-16), that the graph correlation will be some value between -1 and 1; indicating a change in network structure has occurred between time periods.

For example, the adjacency matrix for time period one is:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Deleting the diagonal:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Vectorizing:

$$t\big(vec(G)\big) = (1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0)$$

Similarly for a single edge flipping from one dyad to another (upper right plot in figure 1-16), the adjacency matrix for time period two is:

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Deleting the diagonal:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Vectorizing:

$$t(vec(G')) = (0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0)$$

The mean for both $t\big(vec(G)\big)$ and $t\big(vec(G')\big)$ is 0.5. Using equation 1-23 gives:

$$t(vec(\boldsymbol{Y} - \mu\mathbf{1})) = \left[-\frac{1}{2} \quad \frac{1}{2} \quad -\frac{1}{2} \quad -\frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2} \quad -\frac{1}{2} \quad -\frac{1}{2} \quad \frac{1}{2} \quad -\frac{1}{2}\right]$$

$$cov(G, G') = \frac{t(vec(\boldsymbol{Y} - \mu\mathbf{1}))(vec(\boldsymbol{Y} - \mu\mathbf{1}))}{(4^2 - 4 - 1)}$$

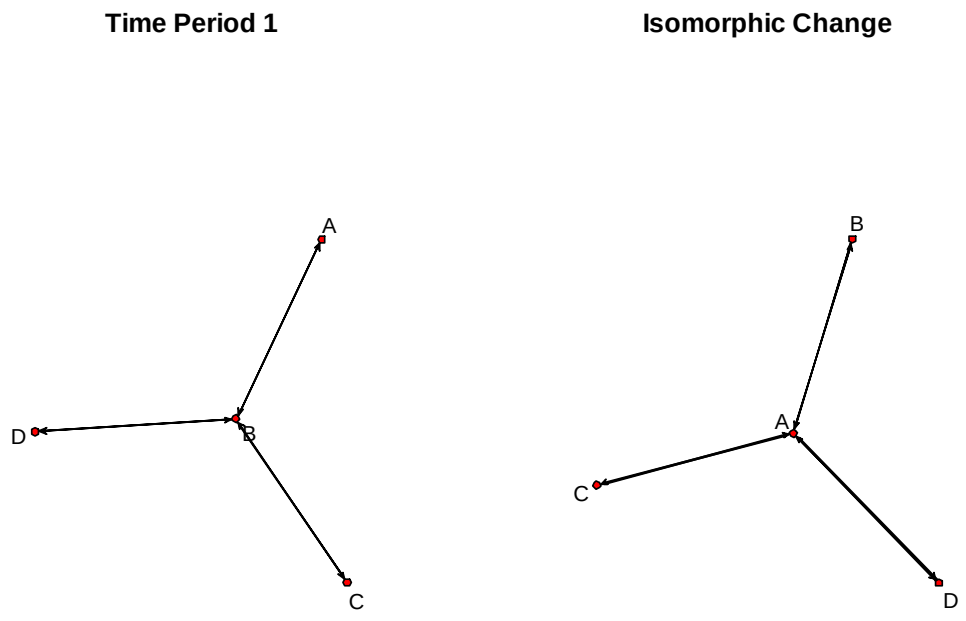
$$= \frac{1}{(11)}\left(-\frac{1}{4} - \frac{1}{4} + \frac{1}{4} - \frac{1}{4} + \frac{1}{4} + \frac{1}{4} - \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4}\right) = \frac{1}{11}$$

The variance for each graph is found in a similar fashion; the variance of both graphs *G* and *G'* is 0.273. Utilizing these three values to calculate the graph correlation gives:

$$\rho(G, G') = \frac{1}{11 * \sqrt{0.273 * 0.273}} = 0.3\bar{3}.$$

Repeating the graph correlation calculation for G = Time Period 1 and G' = Time Period 2B in figure 1-16 yields $\rho(G, G') = 0.707$. Finally, the graph correlation calculation for G = Time Period and G' = Time Period 2C figure 1-16 yields $\rho(G, G') = 0.707$. So what if the change in the network is isomorphic, meaning the structure in time period two is exactly the same as the structure in time period one, only now the node identities have flipped. For example, suppose A and B switch positions during the time between observing time period one and time period two (see figure 1-17).

Figure 1-17: Isomorphic change between time period one and two.



The graph correlation for the isomorphic change is $-0.3\overline{3}$. Based on the four examples presented, it is apparent that the graph correlation is a useful tool for detecting structural changes in a network. The only requirement is for the analyst to keep the order of the nodes consistent from measurement to measurement.

The second method utilizes an exponential random graph model (ERGM). An ERGM is a model of the form:

$P[Y = y|y \in \square] = exp\{\theta' u(y) - \psi(y)\}$ (Equation 4, Snijders, etal, 2004)

Where: $\square$ is the support of Y, the sample space of all possible edges; Y is the random variable representing the presence or absence of an edge and is assumed to have a Bernoulli distribution; y is the realization of Y and can take on values of 1 or 0; θ is a vector of parameters; $u(y)$ is a vector of observed statistics taken from the network in question, and $\psi(y)$ is a normalizing constant that ensures the sum of the probabilities is 1.

Rewriting Snijder, etal's Equation 4 yields the common form of a logistic regression:

$$log\left[\frac{\pi_{ij}(y)}{1-\pi_{ij}(y)}\right] = \theta' u(y)$$

Where: $\pi_{ij}(y)$ is the probability of getting a 1 (i.e. of an edge being present between the i^{th} and j^{th} nodes); and $\frac{\pi_{ij}(y)}{1-\pi_{ij}(y)}$ are the odds of an edge being present between the i^{th} and j^{th}

nodes. Now, taking $u(y)$ as a vector of structural indices (either NLI or GLI) a model can be fit to both *G* and *G'* and compared to determine if a change has occurred.

This method will be highly dependent on the validity of the model being used. Once the model is developed using the observed measures from G, a Monte Carlo simulation is done to simulate M networks from the model, then the indices of the second network are compared to the simulated models to determine if it is reasonable that the second network could be of the same form. Additionally, ERGMs are used to predict network growth (either expansion, or new connections within a closed network); i.e. friends of friends are more likely to become friends.

1.6 Proposal: Detecting Changes in a Network Based on the Number of Communications.

For the purposes of this work it is assumed that the structure of the network remains unchanged from time period to time period. Obviously, in practice, one should test this assumption by utilizing one of the methods developed by the network analysis community, such as graph covariance. This dissertation also assumes that the links or relationships between nodes in a network are already understood and that some method for capturing the total number of communications between nodes connected by an edge exists.

To begin, the concept of a weighted adjacency matrix is introduced. A weighted adjacency matrix is simply the classical adjacency matrix where the 1's have been replaced by the number of communications between the nodes. These matrices are measured at fixed intervals in time (i.e. hourly, daily, weekly, etc.). Next, a multi-dimensional scaling technique is used to obtain a configuration in space (initially this

configuration will be in two dimensions). Utilizing configuration matching techniques the configurations are matched sequentially. For example, the configuration for time period one is matched with the configuration of time period two; time period two with time period three; etc. The configuration matching techniques considered are procrustes, and Euclidean, affine, and projective bi-dimensional regression. Each of these matching techniques yields a coefficient of determination which can be compared to detect a change in network based on the number of communications.

As previously stated, the MDS algorithms and the configuration matching techniques considered are explained in chapter 2. Chapter 2 will also discuss the problem of determining configuration dimension, and extend current two- and three-dimensional configuration matching techniques to k ($k > 3$) dimensions. In chapter 3, a simulation study is presented which demonstrates the technique for detecting a change in communications volume based on an underlying AR1 (autoregressive of order 1) structure between time periods. Finally, chapter 4 will discuss the limitations in the proposed methodology, as well as recommended paths for future work.

1.7 References

S. P. Borgotti, M. G. Everett, and L. C. Freeman (2002). UCINET for Windows: Software for Social Network Analysis. Harvard, MA: Analytic Technologies.

Carter T. Butts, Carley KM (2008). "network: A Package for Managing Relational Data in R." *Journal of Statistical Software*, 24(2), 1-36.

Carter T. Butts, Mark S. Handcock, and David R. Hunter. (July 26, 2008). network: Classes for Relational Data. R package version 1.4-1. Irvine, CA. http://statnet.org/package=network.

Carter T. Butts (2010). sna: Tools for Social Network Analysis. R package version 2.1-0. http://CRAN.R-project.org/package=sna.

J.A. Davis, and S. Leinhardt (1972). The Structure of Positive Interpersonal Relations in Small Groups. *Sociological Theories in Progress*, **2**, 218-251.

P. Eades (1984). A Heuristic for Graph Drawing. *Congressus Numerantium*, 42, 149 – 160.

Katherine Faust (2006). Comparing Social Networks: Size, Density, and Local Structure. *Metodološki zvezki*, 3(2), 185-216.

Thomas M. J. Fruchterman, and Edward Reingold (1991). Graph Drawing by Force-Directed Placement. Software: Practice and Experience, 21(11), 1129-1164.

Tomihisa Kamada, and Satoru Kawai (1989). An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters*, 31(1), 7-15.

Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, and Martina Morris (2003) Software Tools for the Statistical Modeling of Network Data. Version 2.1-1. Project home page at http://statnet.org, URL http://CRAN.R-project.org/package=statnet.

Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, and Martina Morris (2008) "statnet: Software Tools for the representation, Visualization, Analysis and Simulation of Network Data." *Journal of Statistical Software*, 24(1), 1-11.

Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, Martina Morris (2010), ergm: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks. Version 2.2-5. Project home page at http://statnetproject.org. URL http://CRAN.R-project.org/package=ergm.

Robert A Hanneman, Mark Riddle (2005) "Introduction to Social Network Methods." University of California, Riverside. URL http://faculty.ucr.edu/~hanneman.

P.W. Holland, and S. Leinhardt (1976) Local Structure in Social Networks. *Sociological Methodology*, **7**, 1-45.

David R. Hunter, Mark S. Handcock, Carter T. Butts, Steven M. Goodreau, and Martina Morris (2008) "ergm: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks." *Journal of Statistical Software*, 24(3), 1-29.

Martina Morris, Mark S. Handcock, and David R. Hunter (2008) "Specification of Exponential-Family Random Graph Models: Terms and Computational Aspects." *Journal of Statistical Software*, 24(4), 1-24.

R Development Core Team (2009). R: A language and environment for statistical computing. R foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Tom A.B. Snijders, Philippa E. Pattison, Garry L. Robins, Mark S. Handcock (2004). New Specifications for Exponential Random Graph Models. *Sociological Methodology*, 36(1), 99-153.

Chapter 2

Multidimensional Scaling and Configuration Matching

As far back as the 1930's, psychologists have been modeling stimulus response data utilizing matrices of a predetermined minimal rank to approximate matrices of much greater rank. In 1938, Householder and Young developed and proved seven theorems that put a rigorous mathematical foundation beneath the practice of the psychology community. In essence, these theorems show that any matrix of Euclidean distances, ***A,*** can be represented with a series of vectors referenced to the origin of k-space which preserves the Euclidean Distances. This work looks to expand on these ideas viz-a-vis communications networks, treating the number of communications between nodes as a measure of "distance" or dissimilarity.

2.1 Determining Configurations from Distance

Torgerson (1952) extended the work of Householder and Young to include quantitative measures of similarity and dissimilarity (from now on referred to as dissimilarity). Torgerson starts by noting that in order to avoid a floating reference point the matrix of dissimilarities must be converted to a matrix, B referenced to a known point. Torgerson chose the centroid of the coordinate system to be the reference point. Torgerson's method is ultimately based on a spectral value decomposition of an adjusted distance (dissimilarity) matrix that has been centered. Today, Torgerson's method is best known as metric scaling or classical multi-dimensional scaling (CMDS).

In addition to Metric Scaling, a series of non-metric methods have been developed to deal with dissimilarities that are ordinal in nature. The first conceptualization of a non-metric method was proposed by Shepard (1962) in which he ranked and standardized dissimilarities. The first formal, mathematically rigorous non-metric method was put forth by Kruskal (1964). Kruskal refined / improved the work done by Shepard by introducing minimization of a loss function called Stress, S, as an additional iterative step. Today Kruskal's method is best known as Non-metric Least Squares Scaling or simply Least Squares Scaling.

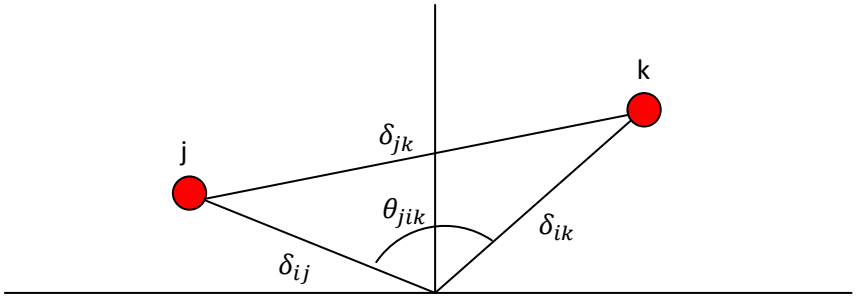
2.1.1 Metric Scaling

The derivation for metric scaling starts by noting that for any two points, the vector from the origin to each point and the line connecting each point makes a triangle. From the cosine law, the cosine of the angle whose vertex is the origin and whose sides are the vectors to each point can be found (equation 2.1).

Cosine Law: $\delta_{jk}^2 = \delta_{ik}^2 + \delta_{ij}^2 - 2\delta_{ik}\delta_{ij}Cos(\theta_{jik})$ (2.1)

where θ_{jik} is the angle opposite side δ_{jk}.

Figure 2-9: Visualization of the Cosine Law.



Rearranging (2.1) gives:

$$Cos(\theta_{jik}) = \frac{\delta_{ik}^2 + \delta_{ij}^2 - \delta_{jk}^2}{2\delta_{ik}\delta_{ij}} \quad (2.2)$$

Torgerson then defines the scalar product associated with the dissimilarity between j and k as:

$$b_{jk} = \frac{\delta_{ik}^2 + \delta_{ij}^2 - \delta_{jk}^2}{2} \quad (2.3)$$

$$b_{jk} = \delta_{ik}\delta_{ij}Cos(\theta_{jik}) \quad (2.4)$$

The following derivation relies heavily on lecture notes written by Forrest W Young, Professor Emeritus, Quantitative Psychology, University of North Carolina. Professor Young's lectures can be found at http://forrest.psych.unc.edu.

First, define the matrix of scalar products, $\boldsymbol{B}$, whose elements are the b_{jk} defined in equation 2.4. Note, the definition of $\boldsymbol{B}$ implies that the reference point for each b_{jk} is the centroid of the Cartesian coordinates of the n points. Now suppose that an nxk matrix, $\boldsymbol{X}$, exists such that $\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}'$ and where $\boldsymbol{X}$, is the matrix of Cartesian coordinates for each of the n points of interest. Torgerson defines: $\boldsymbol{X}^* = \boldsymbol{X} - \boldsymbol{J}\boldsymbol{c}$, where $\boldsymbol{J}$ is an $nx1$ matrix of 1's, and $\boldsymbol{c} = \frac{\boldsymbol{J}'\boldsymbol{X}}{n}$. Here, translating $\boldsymbol{X}$ to $\boldsymbol{X}^*$, ensures that the coordinates of each point (one set of coordinates is a row in $\boldsymbol{X}^*$) are referenced to the centroid of the points. Now, let $\boldsymbol{B}^* = \boldsymbol{X}^*\boldsymbol{X}^{*'}$. Plugging in $\boldsymbol{X} - \boldsymbol{J}\boldsymbol{c}$ for $\boldsymbol{X}^*$ and expanding gives:

$$\boldsymbol{B}^* = \boldsymbol{X}^*\boldsymbol{X}^{*'} = (\boldsymbol{X} - \boldsymbol{J}\boldsymbol{c})(\boldsymbol{X} - \boldsymbol{J}\boldsymbol{c})' = (\boldsymbol{X} - \boldsymbol{J}\boldsymbol{c})(\boldsymbol{X}' - \boldsymbol{c}'\boldsymbol{J}')$$

$$= \boldsymbol{XX'} - \frac{1}{n}\boldsymbol{XX'JJ'} - \frac{1}{n}\boldsymbol{JJ'XX'} + \frac{1}{n^2}\boldsymbol{JJ'XX'JJ'}$$

$$= \boldsymbol{B} - \frac{1}{n}\boldsymbol{BJJ'} - \frac{1}{n}\boldsymbol{JJ'B} + \frac{1}{n^2}\boldsymbol{JJ'BJJ'}$$

Notice here that $\frac{1}{n}\boldsymbol{BJJ'} = row\ means\ of\ \boldsymbol{B}$; $\frac{1}{n}\boldsymbol{JJ'B} = column\ means\ of\ \boldsymbol{B}$, and $\frac{1}{n^2}\boldsymbol{JJ'BJJ'} = the\ grand\ mean\ of\ all\ the\ elements\ in\ \boldsymbol{B}$.

Isolating a single element in $\boldsymbol{B}^*$ gives:

$$b_{ij}^* = b_{ij} - \frac{1}{n}\sum_{l=1}^{n} b_{il} - \frac{1}{n}\sum_{m=1}^{n} b_{mj} + \frac{1}{n^2}\sum_{g=1}^{n}\sum_{h=1}^{n} b_{gh} \quad (2.5)$$

Substituting in equation 2.3 for b_{ij} and simplifying gives:

$$b_{ij} = -\frac{1}{2}\left[\delta_{ij}^2 - \overline{\delta_{i\cdot}^2} - \overline{\delta_{\cdot j}^2} + \overline{\delta_{\cdot\cdot}^2}\right] \quad (2.6)$$

Therefore, given a matrix, $\boldsymbol{D}$, of Euclidean distances, or $\boldsymbol{\Delta}$, the measures of dissimilarity, the matrix of scalar products $\boldsymbol{B}$, can be found by

1) Squaring each element in $\boldsymbol{D}$ ($\boldsymbol{\Delta}$); call the matrix of squared elements $\boldsymbol{D^2}$ ($\boldsymbol{\Delta^2}$).
2) Find the row means, column means, and grand mean of $\boldsymbol{D^2}$ ($\boldsymbol{\Delta^2}$).
3) Find the centering matrix: $\boldsymbol{C} = -\boldsymbol{M}(row\ means) - \boldsymbol{M}(col\ means) + \boldsymbol{M}(grand\ means)$
4) Solve for $\boldsymbol{B} = \boldsymbol{D^2} + \boldsymbol{C}$.

Now, if the matrix $\boldsymbol{D}$ is made up of Euclidean distances, then $\boldsymbol{D}$ is square and symmetric, which implies that $\boldsymbol{B}$ will also be square and symmetric. From matrix algebra, spectral

decomposition can be used to decompose a symmetric, square matrix into three matrices $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{U'}$, such that $\boldsymbol{B} = \boldsymbol{UVU}'$. Here the columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{B}$ and $\boldsymbol{V}$ is a diagonal matrix of eigenvalues of $\boldsymbol{B}$. If $\boldsymbol{B}$ is positive definite, the elements on the diagonal of $\boldsymbol{V}$ will be strictly positive; if $\boldsymbol{B}$ is nonnegative definite, the elements on the diagonal of $\boldsymbol{V}$ will be greater than or equal to 0. In cases where $\boldsymbol{B}$ is either positive definite or nonnegative definite, the matrix V of eigenvalues can be rewritten as $\boldsymbol{V} = \boldsymbol{V}^{0.5}\boldsymbol{V}^{0.5}$. Now $\boldsymbol{B} = \boldsymbol{XX}' = \boldsymbol{UVU}' = (\boldsymbol{UV}^{0.5})(\boldsymbol{V}^{0.5}\boldsymbol{U}')$. In essence, the matrix of coordinates, $\boldsymbol{X}$, implied by the matrix of distances (dissimilarities) is simply $\boldsymbol{UV}^{0.5}$.

Recall that the motivation for Torgerson, was to find coordinates in low (say $k << n$) dimension Euclidean space such that the absolute distances from n-dimensional space are preserved as much as possible. This implies the need for a way to determine the required number of dimensions. Torgerson's solution is exactly the solution used in principle component analysis; the ratio of the sum of the first k eigenvalues to the sum of all n eigenvalues is compared to a threshold value. The k for which the ratio is greater than, or equal to the threshold value is the reduced dimension. It is important to remember that eigenvalues are not necessarily distinct, and some reduction in dimension may occur due to possible non-distinct eigenvalues.

For example, consider the 4 x 4 weighted adjacency matrix:

$$\boldsymbol{D} = \begin{bmatrix} 0 & 55 & 30 & 40 \\ - & 0 & 85 & 90 \\ - & - & 0 & 35 \\ - & - & - & 0 \end{bmatrix}$$

Squaring each element of $\boldsymbol{D}$ above, double centering, and multiplying by -0.5 yields, $\boldsymbol{B}$:

$$\boldsymbol{D^2} = \begin{bmatrix} 0 & 3025 & 900 & 1600 \\ - & 0 & 7225 & 8100 \\ - & - & 0 & 1225 \\ - & - & - & 0 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} 1.5625 & 92.1875 & 29.6875 & -123.4375 \\ - & 3207.8125 & -1529.6875 & -1770.3125 \\ - & - & 957.8125 & 542.1875 \\ - & - & - & 1351.5625 \end{bmatrix}$$

The eigen analysis of B was carried out using the eigen function in R. This analysis yielded four eigenvalues; one of which is negative meaning that B is an indefinite matrix.

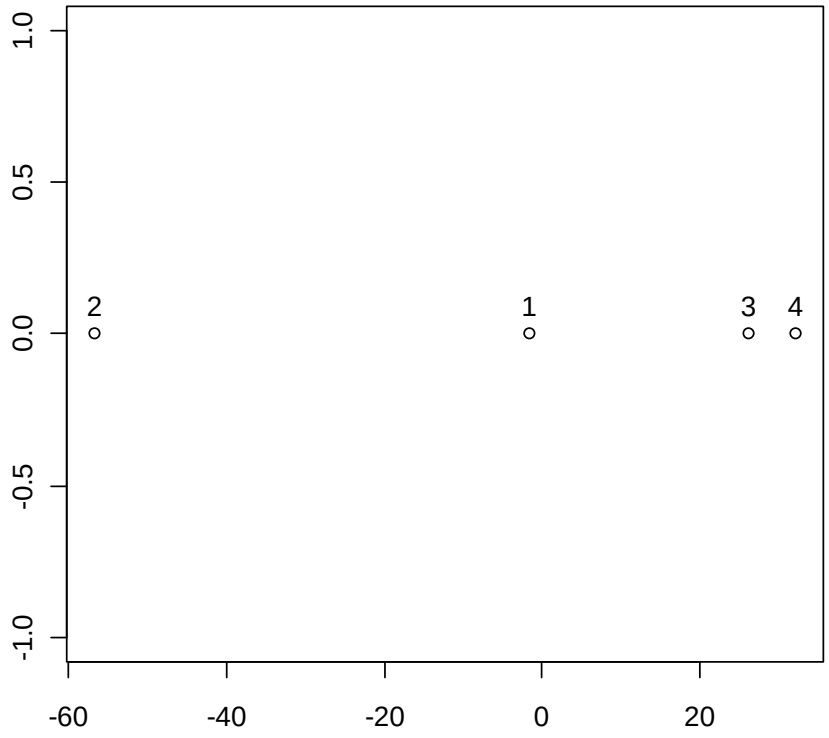
$$V = \begin{bmatrix} 4922.87 & 0 & 0 & 0 \\ 0 & 619.74 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -23.86 \end{bmatrix}$$

In this situation, Torgerson suggests only focusing on the nonnegative eigenvalues, so the test for dimensionality will be based on the ratio of the sum of the first k nonnegative eigenvalues to the total of the nonnegative eigenvalues. Using a threshold of 0.8 yields a k of 1:

$$\frac{\sum_{i=1}^{1} \lambda_i}{\sum_{j=1}^{2} \lambda_j} = \frac{4922.87}{4922.87 + 619.74} = 0.888$$

The resulting plot:

Figure 2-10: Plot of Torgerson's Configuration in one dimension.



The resulting distances in 1 dimension are:

$$\widehat{\boldsymbol{D}_{\mathbf{1D}}} = \begin{bmatrix} 0 & 54.89 & 27.92 & 33.80 \\ - & 0 & 82.81 & 88.69 \\ - & - & 0 & 5.88 \\ - & - & - & 0 \end{bmatrix}; \; \boldsymbol{D}_{\mathbf{1D}} = \begin{bmatrix} 0 & 55 & 30 & 40 \\ - & 0 & 85 & 90 \\ - & - & 0 & 35 \\ - & - & - & 0 \end{bmatrix}$$

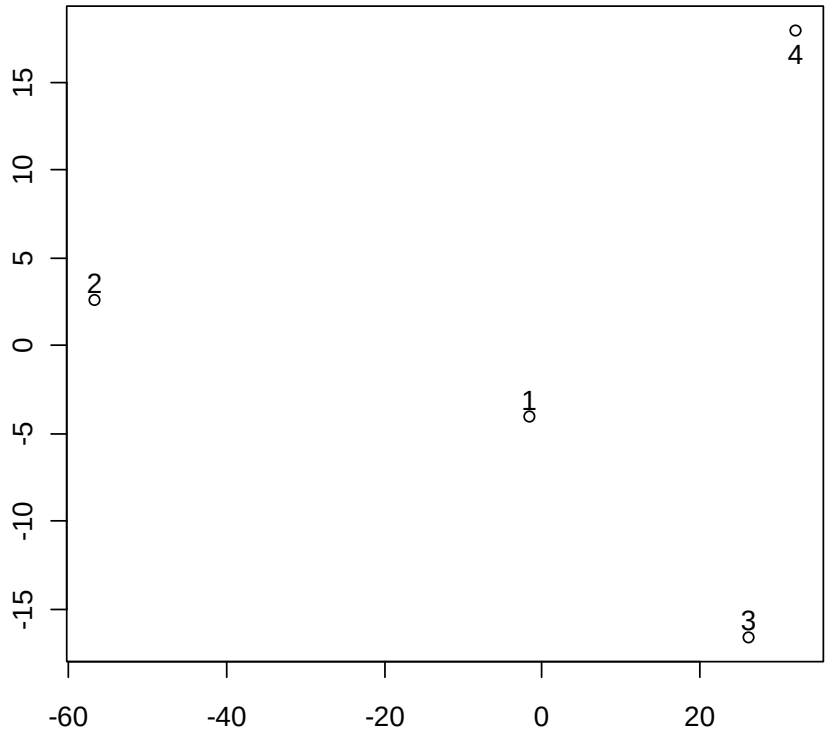
The residual distance in the one dimensional configuration is:

$$\boldsymbol{D_{1D}} - \widehat{\boldsymbol{D_{1D}}} = \begin{bmatrix} 0 & 0.11 & 2.08 & 6.20 \\ - & 0 & 2.19 & 1.31 \\ - & - & 0 & 29.12 \\ - & - & - & 0 \end{bmatrix}$$

Giving a raw stress: $RS_{1D} = \sqrt{\left\{\left(\boldsymbol{D_{1D}} - \widehat{\boldsymbol{D_{1D}}}\right)'\left(\boldsymbol{D_{1D}} - \widehat{\boldsymbol{D_{1D}}}\right)\right\}} = 29.95.$

However, since there are only two positive eigenvalues it makes sense to plot the configuration in two dimensions.

Figure 2-11: Torgerson's Configuration in Two Dimensions.



Yielding configuration distances:

$$\widehat{\boldsymbol{D}_{\mathbf{2D}}} = \begin{bmatrix} 0 & 55.29 & 30.62 & 40.29 \\ - & 0 & 85.00 & 90.00 \\ - & - & 0 & 35.02 \\ - & - & - & 0 \end{bmatrix}; \; \boldsymbol{D}_{\mathbf{2D}} = \begin{bmatrix} 0 & 55 & 30 & 40 \\ 55 & 0 & 85 & 90 \\ 30 & 85 & 0 & 35 \\ 40 & 90 & 35 & 0 \end{bmatrix}$$

The residual distance in the two dimensional configuration is:

$$\boldsymbol{D}_{\mathbf{2D}} - \widehat{\boldsymbol{D}_{\mathbf{2D}}} = \begin{bmatrix} 0 & -0.29 & -0.62 & -0.29 \\ - & 0 & 0 & 0 \\ - & - & 0 & -0.02 \\ - & - & - & 0 \end{bmatrix}$$

Giving a raw stress: $RS_{2\mathrm{D}} = \sqrt{\left\{\left(\boldsymbol{D}_{\mathbf{2D}} - \widehat{\boldsymbol{D}_{\mathbf{2D}}}\right)'\left(\boldsymbol{D}_{\mathbf{2D}} - \widehat{\boldsymbol{D}_{\mathbf{2D}}}\right)\right\}} = 0.74$.

From inspection of the residual distances for both the one and two dimensional cases, the two dimensional case does a much better job of capturing the original configuration vis-à-vis the distances.

2.1.2 Non-metric Least Squares Scaling

As stated previously, the first formal non-metric method was put forth by Kruskal (1964) with the introduction of a loss function, called stress and a step which minimizes the same. There are two definitions of stress that are commonly used in non-metric scaling the first, Stress1 (2.6) was the original stress defined by Kruskal. Later, due to Stress1 being very small, a second loss function called, Stress2, (2.7) was defined. Stress 2 gives rise to larger stress values, thus allowing the algorithm to further refine the node positions. In the example to follow, Stress2 vice Stress1 is used to demonstrate Kruskal's non-metric scaling algorithm.

$$Stress1 = \sqrt{S^*/_{T^*}}; where\ S^* = \sum_i \sum_j (d_{ij} - \widehat{d_{ij}})^2; and\ T^* = \sum_i \sum_j (d_{ij})^2 \quad (2.6)$$

$$Stress2 = \sqrt{S^*/_{T'}}; where\ S^* = \sum_i \sum_j (d_{ij} - \widehat{d_{ij}})^2; and\ T' = \sum_i \sum_j (d_{ij} - d_{..})^2 \quad (2.7)$$

Kruskal's Algorithm:

1. Rank the dissimilarities in ascending order and index these ranked dissimilarities 1 through k, where $k = (n^2 - n)/2$.
    a. Example: Consider four points labeled 1, 2, 3, 4 with weighted adjacency matrix, $\boldsymbol{M}$, where:

$$\boldsymbol{M} = \begin{bmatrix} 0 & 55 & 30 & 40 \\ 55 & 0 & 85 & 90 \\ 30 & 85 & 0 & 35 \\ 40 & 90 & 35 & 0 \end{bmatrix}$$

    This gives the set of sorted dissimilarities:

$$\delta_{13} = 30;\ \delta_{34} = 35;\ \delta_{14} = 40;\ \delta_{12} = 55;\ \delta_{23} = 85;\ \text{and } \delta_{24} = 90$$

    Re-indexing gives:

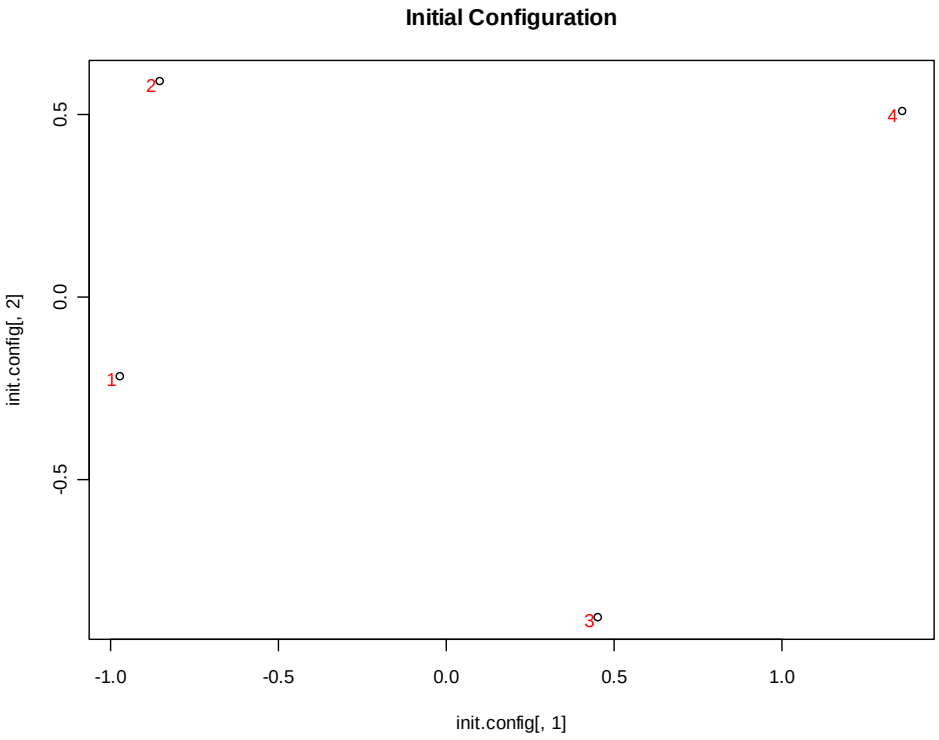
$$\delta_1 = 30;\ \delta_2 = 35;\ \delta_3 = 40;\ \delta_4 = 55;\ \delta_5 = 85; \text{and } \delta_6 = 90$$

2. Choose a starting configuration. If a random process is used to choose the initial configuration, then Kruskal recommends centering and normalizing the points, to prevent the algorithm from "wandering" around the plane. Typically the classical MDS configuration is chosen as the starting point, and then only normalization of the points is required. However, to illustrate how the algorithm works, an ad hoc

configuration was generated by randomly placing four points on a -2 to 2 square grid and estimating the associated coordinate for the four node example.

Example starting configuration: node 1: (-0.97, -0.22); node 2: (-0.85, 0.59); node 3: (0.46, -0.88); node 4: (1.36, 0.5), see figure 2-4.

Figure 2-12: Initial starting configuration.



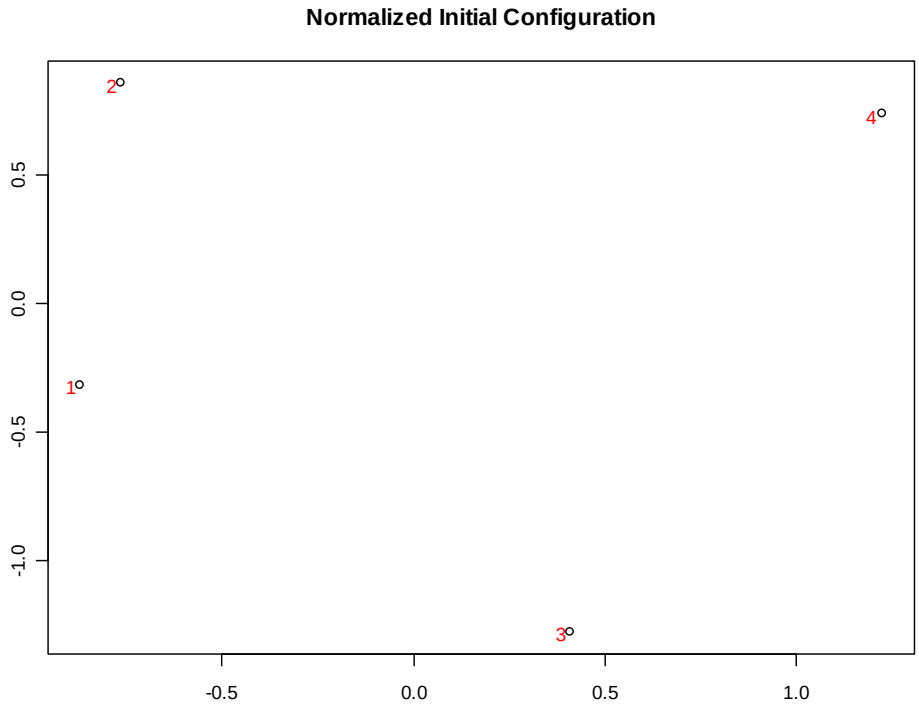
b. Normalize the initial coordinates: The average of the X and Y values are essentially 0 (2.5×10^{-8} and 2.1×10^{-17}, respectively); the standard deviation of the X and Y values is 1.11 and 0.69, respectively. Normalizing the coordinates yields the following coordinate matrix:

$$Normalized\ Initial\ Coordinates = \begin{bmatrix} -0.87 & -0.32 \\ -0.76 & 0.86 \\ 0.41 & -1.27 \\ 1.22 & 0.74 \end{bmatrix}$$

Giving initial distances (equation 2.6):

$d_{12} = 1.18;\ d_{13} = 1.60;\ d_{14} = 2.34;\ d_{23} = 2.43\ ;\ d_{24} = 1.99\ ;\ d_{34} = 2.17$

Figure 2-13: Normalized Initial Configuration.

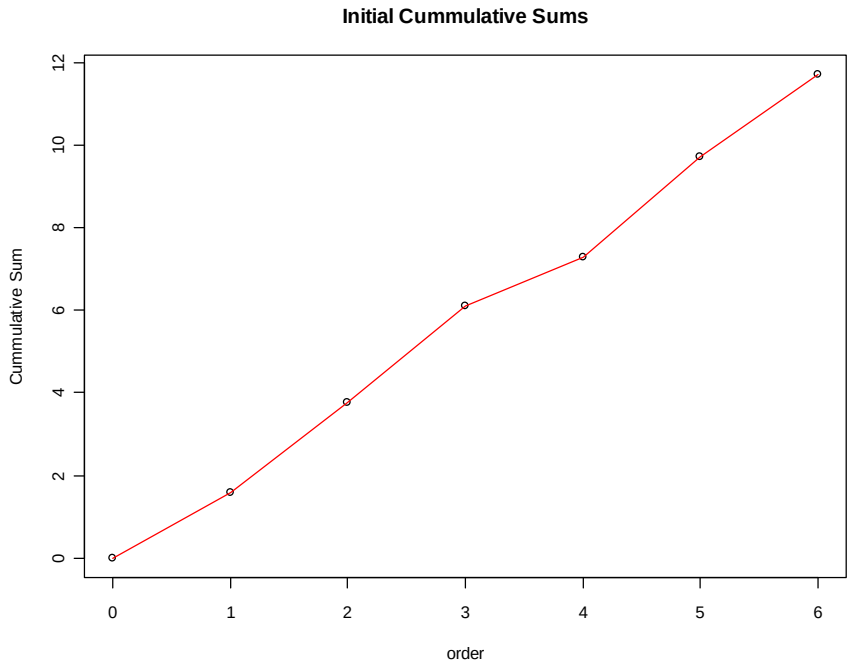


3. Sort the d_{ij} by preserving the order of the ranked dissimilarities.
    a. Recall in step 1, the original dissimilarities were re-indexed into "i" notation where $i = \{1, 2, 3, 4, 5, 6\}$. Sorting the d_{ij} from step 2 yields:

$$d_1 = 1.60;\ d_2 = 2.17;\ d_3 = 2.34;\ d_4 = 1.18;\ d_5 = 2.43\ ;\ d_6 = 1.99$$

4. Use isotonic regression to regress the initial d_{ij} onto the order of the δ_k. Isotonic regression is a regression method, that allows the user to take into account order if there happened to be a priori information that say $(x_1) \leq \mu(x_2) \leq \cdots \leq \mu(x_n)$. The most common method of performing isotonic regression is to use the *Pool-Adjacent-Violators* (Barlow, et al 1972) algorithm. Graphically, isotonic regression is the same as finding the greatest convex minorant (GCM) for the plot of cumulative sums of distance versus rank order. Figure 2-6 is a plot of cumulative sums for the distances derived from the initial, normalized configuration.

Figure 2-14: Cumulative sums versus rank order.



From figure 2-6 it is easy to identify the adjacent violators by comparing the slopes of the line segments between points. Going from left to right, if the slope of each line segment stays the same or gets steeper, then $d_i \leq d_j; i < j$. However, if the slope of an adjacent line segment is shallower than the slope of the previous line segment then $d_i > d_j; i < j$ and a violation has occurred.

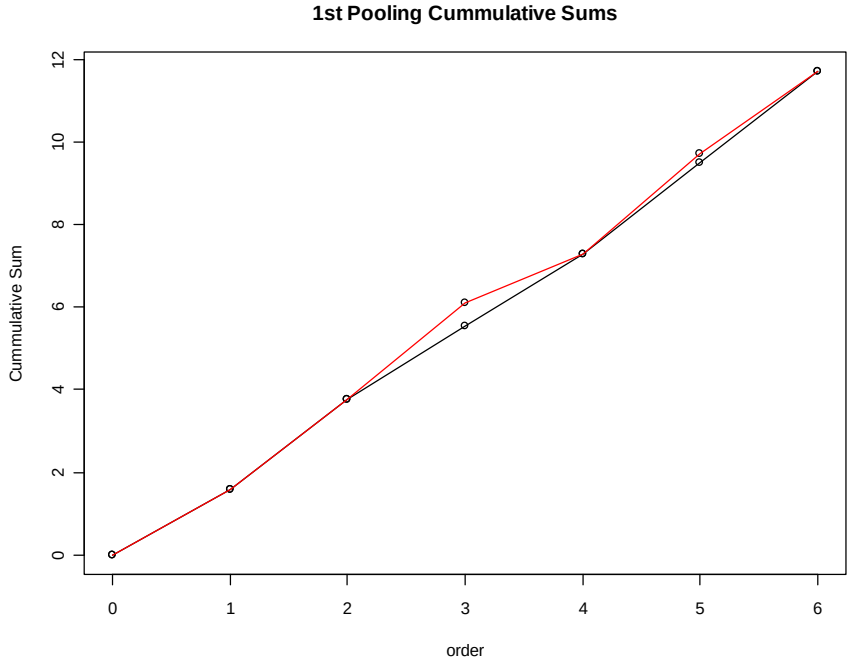
a. In the current example, the δ_i (from step 1) play the role of $\mu(x_i)$; and the d_i (from step 2) the role of the response variable from ordinary least squares regression (OLS).
b. The *Pool-Adjacent-Violators* algorithm starts with the ordered distances. A violator is any value that is less than the value to its immediate left. Start by searching from right left. If $d_1 \leq d_2$there is no violation and the values are not pooled; however, if $d_1 > d_2 \leq d_3$ then $\widehat{d_1} = \widehat{d_2} = {(d_1 + d_2)}/{2}$.

Table 2-1: First pool-Adjacent violators are highlighted in yellow.

i	1	2	3	4	5	6
d_i	1.60	2.17	2.34	1.18	2.43	1.99
1st pool	1.60	2.17	$\frac{2.34 + 1.18}{2}$	$\frac{2.34 + 1.18}{2}$	$\frac{2.43 + 1.99}{2}$	$\frac{2.43 + 1.99}{2}$
$\widehat{d_i}$	1.60	2.17	1.76	1.76	2.21	2.21

After the first pooling of adjacent violators, there remains a violation between positions 2 and 3 (See Figure 2-7).

Figure 2-15: Cumulative sums after the first pooling versus order.



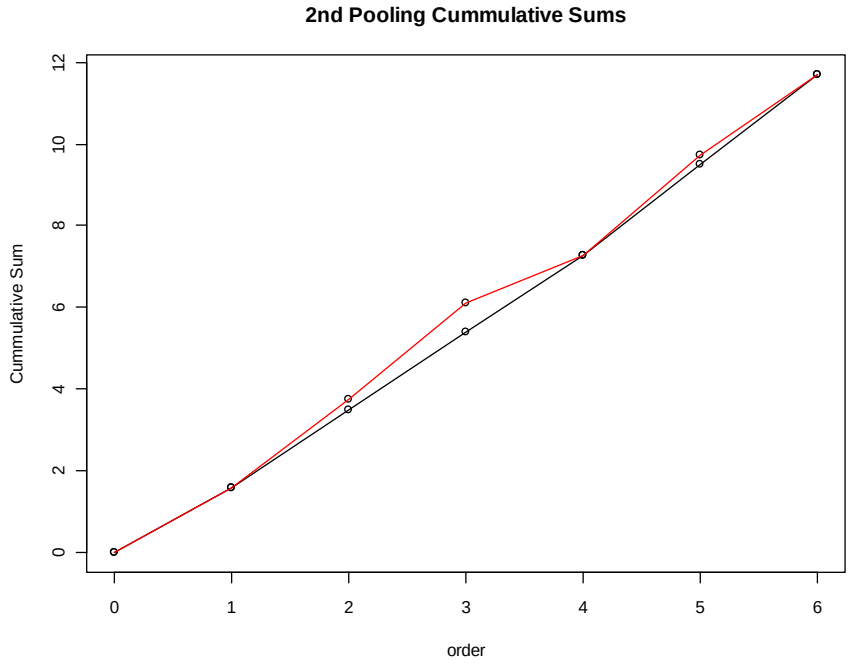
Therefore, a second pooling is required.

Table 2-2: Second pool - Adjacent violators are highlighted in yellow.

i	1	2	3	4	5	6
d_i	1.60	2.17	1.76	1.76	2.21	2.21
2^{nd} pool	1.60	$\frac{2.17 + 2.34 + 1.18}{3}$			2.21	2.21
$\hat{d}_i$	1.60	1.90	1.90	1.90	2.21	2.21

Recall, that from the first pooling the value in position 3 is an average of two values necessitating that the second pooling will be done by averaging the three values $d_2, d_3,$ and d_4. Notice here that there are no longer any adjacent violations, and thus the $\hat{d}_i$ in Table 3, minimize the stress in equations 2.6 and 2.7. Figure 2-8 shows the GCM in black and the original distances in red.

Figure 2-16: Cumulative distances for the final pooling versus order.



5. The next step in the algorithm is to find the gradient of the stress and compare it to a predetermined small value, ϵ. If $\left|\frac{dS}{dx}\right| < \epsilon$, then the algorithm stops and the current configuration has the minimum stress, and therefore matches the true configuration as well as possible for the given number of dimensions. If $\left|\frac{dS}{dx}\right| \geq \epsilon$, the algorithm continues on to the next step, which is to perturb each coordinate and then go back to step 2 with the new configuration. In practice, ϵ is set between 10^{-6} and 10^{-8}. For this example, $\epsilon = 10^{-6}$is used.

   Recall, that Stress2 is being used in this example; the gradient with respect to each x_{ui} of Stress2 is:

$$\frac{\partial S}{\partial x_{ui}} = \frac{1}{2}\left\{\frac{1}{S}\right\}\frac{1}{\sum_{r,s}(d_{rs} - d_{..})^2}\left\{\frac{\partial \sum_{r,s}\left(d_{rs} - \hat{d}_{rs}\right)^2}{\partial x_{ui}} - S^2\frac{\partial \sum_{r,s}(d_{rs} - d_{..})^2}{\partial x_{ui}}\right\} \quad (2.8)$$

where x_{ui} is an element of the coordinate vector $\boldsymbol{x}$

$$= \{x_{11}, .., x_{14}, x_{21}, \dots, x_{24}\}$$

$$u = 1 \text{ if the x} - \text{coordinate of the i}^{\text{th}} \text{ node;}$$

$$u = 2 \text{ if the y} - \text{coordinate of the i}^{\text{th}} \text{ node;}$$

$$i = 1, 2, 3, \dots, n; n = \text{number of nodes.}$$

Now let:

$$S^* = \sum_{r,s}\left(d_{rs} - \hat{d}_{rs}\right)^2 \quad (2.9)$$

And,

$$T' = \sum_{r,s}(d_{rs} - d_{..})^2 \quad (2.10)$$

Then, from vector calculus:

$$\frac{dS^*}{d\boldsymbol{x}} = \left\{\frac{\partial S^*}{\partial x_{11}}\frac{\partial S^*}{\partial x_{12}}\frac{\partial S^*}{\partial x_{13}}\frac{\partial S^*}{\partial x_{14}}\frac{\partial S^*}{\partial x_{21}}\frac{\partial S^*}{\partial x_{22}}\frac{\partial S^*}{\partial x_{23}}\frac{\partial S^*}{\partial x_{24}}\right\} \quad (2.11)$$

$$\frac{\partial S^*}{\partial x_{ui}} = 2\sum_{i \neq s}\left(d_{is} - \hat{d}_{is}\right)\left(\frac{\partial d_{is}}{\partial x_{ui}}\right) \quad (2.11a)$$

$$\frac{\partial d_{is}}{\partial x_{ui}} = \frac{(x_{ui} - x_{us})}{d_{is}} \quad (2.11b)$$

And

$$\frac{dT'}{d\boldsymbol{x}} = \left\{\frac{\partial T'}{\partial x_{11}}\frac{\partial T'}{\partial x_{12}}\frac{\partial T'}{\partial x_{13}}\frac{\partial T'}{\partial x_{14}}\frac{\partial T'}{\partial x_{21}}\frac{\partial T'}{\partial x_{22}}\frac{\partial T'}{\partial x_{23}}\frac{\partial T'}{\partial x_{24}}\right\} \quad (2.12)$$

$$\frac{\partial T^*}{\partial x_{ui}} = 2\sum\nolimits_{i\neq s}\left(d_{is} - \hat{d}_{is}\right)\left(\frac{\partial(d_{is} - \hat{d}_{is})}{\partial x_{ui}}\right) \quad (2.12a)$$

$$\frac{\partial(d_{is} - \hat{d}_{is})}{\partial x_{ui}} = \frac{5}{6}\left(\frac{x_{ui} - x_{us}}{d_{is}}\right) \quad (2.12b)$$

$$\frac{\partial T^*}{\partial x_{ui}} = \frac{5}{3}\sum\nolimits_{i\neq s}\left(d_{is} - \hat{d}_{is}\right)\left(\frac{x_{ui} - x_{us}}{d_{is}}\right) \quad (2.13)$$

It is obvious from equation 2.8, that the stress of the current configuration is needed to find the gradient. This can be done by finding S^* (Table 4)and T' (Table 5) using equations 2.9 and 2.10, respectively.

Table 2-3: Calculation of S*.

d_{rs}	$\hat{d}_{rs}$	$d_{rs} - \hat{d}_{rs}$	$\left(d_{rs} - \hat{d}_{rs}\right)^2$
1.60	1.60	0.00	0.00
2.17	1.90	0.27	0.07
2.34	1.90	0.45	0.20
1.18	1.90	-0.72	0.52
2.43	2.21	0.22	0.05
1.99	2.21	-0.22	0.05
		$S^* = \sum_{r,s}\left(d_{rs} - \hat{d}_{rs}\right)^2 =$	0.89

The grand mean of the distances is: $d_{..} = \sum_{r,s} d_{rs}/(n(n-1)/2)$

$$d_{\cdot\cdot} = \frac{1.60 + 2.17 + 2.34 + 1.18 + 2.43 + 1.99}{6} = 1.95$$

Table 2-4: Calculation of T'.

d_{rs}	$d_{\cdot\cdot}$	$d_{rs} - d_{\cdot\cdot}$	$(d_{rs} - d_{\cdot\cdot})^2$
1.60	1.95	-0.35	0.1225
2.17	1.95	0.22	0.0484
2.34	1.95	0.39	0.1521
1.18	1.95	-0.77	0.5929
2.43	1.95	0.48	0.2304
1.99	1.95	0.04	0.0016
		$T' = \sum_{r,s} (d_{rs} - d_{\cdot\cdot})^2 =$	1.15

Putting S* and T' together in equation 2.7 gives:

$$S = \sqrt{S^*/T'} = \sqrt{0.89/1.15} = \sqrt{0.7739} = 0.88$$

Now calculate the partial derivatives of S* and T' with respect to x_{ui}. To demonstrate how the partial with respect to x_{ui} is calculated, only the partial with respect to x_{11} is fully worked out; the remaining 7 partials were calculated using a computer program and are listed following the x_{11} example.

$$\frac{\partial S^*}{\partial x_{11}} = 2\left\{(d_{12} - \hat{d}_{12})\left(\frac{x_{11} - x_{12}}{d_{12}}\right) + (d_{13} - \hat{d}_{13})\left(\frac{x_{11} - x_{13}}{d_{13}}\right) + (d_{14} - \hat{d}_{14})\left(\frac{x_{11} - x_{14}}{d_{14}}\right)\right\}$$

$$\frac{\partial S^*}{\partial x_{11}} = 2\left\{(1.18 - 1.9)\left(\frac{-0.87 - (-0.76)}{1.18}\right) + (1.6 - 1.6)\left(\frac{-0.87 - 0.41}{1.6}\right) + (2.34 - 1.9)\left(\frac{-0.87 - 1.22}{2.34}\right)\right\}$$

$$\frac{\partial S^*}{\partial x_{11}} = -0.357$$

The remaining seven partials are:

$$\frac{\partial S^*}{\partial x_{12}} = 0.99;\ \frac{\partial S^*}{\partial x_{13}} = 0.01;\ \frac{\partial S^*}{\partial x_{14}} = 0.56;\ \frac{\partial S^*}{\partial x_{21}} = 1.80;\ \frac{\partial S^*}{\partial x_{22}} = -1.07;\ \frac{\partial S^*}{\partial x_{23}} = -0.89;\ \frac{\partial S^*}{\partial x_{24}} = 0.93$$

Then equation 2.11 has a value of $\frac{dS^*}{d\boldsymbol{x}} = -0.0003$.

Similarly, the partial of T' with respect to x_{11} is:

$$\frac{\partial T'}{\partial x_{11}} = \frac{5}{3}\left\{(d_{12} - \hat{d}_{12})\left(\frac{x_{11} - x_{12}}{d_{12}}\right) + (d_{13} - \hat{d}_{13})\left(\frac{x_{11} - x_{13}}{d_{13}}\right) + (d_{14} - \hat{d}_{14})\left(\frac{x_{11} - x_{14}}{d_{14}}\right)\right\}$$

$$\frac{\partial T'}{\partial x_{11}} = \frac{5}{3}\left\{(1.18 - 1.9)\left(\frac{-0.87 - (-0.76)}{1.18}\right) + (1.6 - 1.6)\left(\frac{-0.87 - 0.41}{1.6}\right) + (2.34 - 1.9)\left(\frac{-0.87 - 1.22}{2.34}\right)\right\}$$

$$\frac{\partial T'}{\partial x_{11}} = -0.30$$

The remaining seven partials are:

$$\frac{\partial T'}{\partial x_{12}} = 0.82;\ \frac{\partial T'}{\partial x_{13}} = 0.01;\ \frac{\partial T'}{\partial x_{14}} = 0.47;\ \frac{\partial T'}{\partial x_{21}} = 1.50;$$

$$\frac{\partial T^{'}}{\partial x_{22}} = -0.89;\ \frac{\partial T^{'}}{\partial x_{23}} = -0.74;\ \frac{\partial T^{'}}{\partial x_{24}} = 0.78$$

Then equation 2.13 has a value of $\frac{dT^{'}}{dx} = -0.000069$.

Plugging the values for $T^{'}$, S, $\frac{dT^{'}}{dx}$ and $\frac{dS^{*}}{dx}$ into equation 2.8 gives a gradient of -1.2 x 10^{-4} . The magnitude of the gradient is greater ϵ; therefore the algorithm continues to step 6.

6. The sixth step of the algorithm is where the step length to be applied to each coordinate is calculated. To complete this step an initial step length must be selected. The initial step length for this example is 0.1.

   There are three new terms that Kruskal introduces in this step; each was empirically derived. They are the angle factor; relaxation factor; and the good luck factor.

   $$Angle\ Factor = AF = 4^{cos^3\theta};$$

   where $\theta =$ angle between current and previous gradients.

   $$Relaxation\ Factor = RF = \frac{1.3}{1 + (5\ step\ ratio)^5}$$

   $$\text{where } 5\ step\ ratio = \ min\left\{1, \left(\frac{current\ stress}{stress\ 5\ iterations\ ago}\right)\right\}$$

   $$Goodluck\ Factor = GF = min\left\{1, \left(\frac{current\ stress}{previous\ stress}\right)\right\}$$

   For the first four iterations, the 5 step ratio is always 1; for the first iteration, the Goodluck Factor is 1; and the previous and current gradients are coincident, therefore the Angle Factor is 4.

Now calculate the current step length using equation 2.14.

$$sl_{current} = sl_{previous} * (AF) * (RF) * (GF) \quad (2.14)$$

For this example the current step length then becomes:

$$sl_{current} = 0.1 * (4) * (0.65) * (1) = 0.26$$

7. The last step is to find the new configuration as follows:

$$x_{ui}^{j+1} = x_{ui}^{j} - sl_{current} \frac{\frac{\partial S}{\partial x_{ui}}}{\left|\frac{\partial S}{\partial x_{ui}}\right|} \quad (2.15)$$

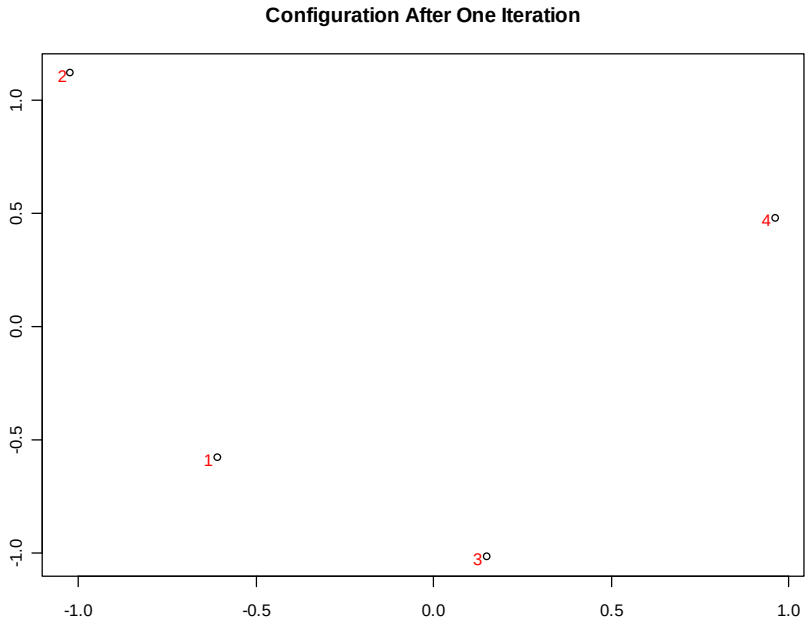
Applying equation 2.15 to each x_{ui} yields the following matrix of new coordinates:

$$\begin{bmatrix} x_{11}^2 & x_{21}^2 \\ x_{12}^2 & x_{22}^2 \\ x_{13}^2 & x_{23}^2 \\ x_{14}^2 & x_{24}^2 \end{bmatrix} = \begin{bmatrix} -0.87 & -0.32 \\ -0.76 & 0.86 \\ 0.41 & -1.27 \\ 1.22 & 0.74 \end{bmatrix} - 0.26 * \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix}$$

$$\text{New Configuration} = \begin{bmatrix} -0.61 & -0.58 \\ -1.02 & 1.12 \\ 0.15 & -1.01 \\ 0.96 & 0.48 \end{bmatrix}$$

The new configuration is plotted in figure 2-9. The algorithm repeats starting with step 2, using the new configuration.

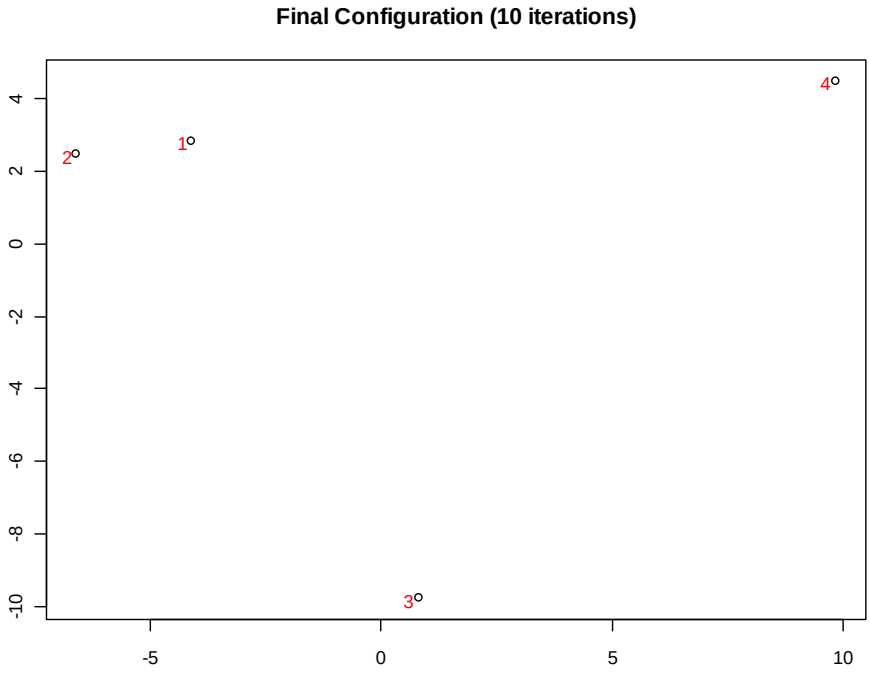
Figure 2-17: Plot of the new configuration after a single iteration of Kruskal's non-metric MDS algorithm.



Notice here that the relative positions of nodes 2, 3, and 4 have not changed, but that node 1 has made a pronounced move towards the center of plot.

The above example was run in the R environment using the isoMDS function in the MASS package. The initial configuration was used to seed the isoMDS function; it took a total of 10 iterations to achieve a gradient < ϵ. The final configuration is plotted in figure 2-10.

Figure 2-18: The final configuration for the 4 node example after 10 iterations of Kruskal's non-metric MDS algorithm.



2.2 Methods for Matching Two-dimensional Configurations

In section 2.1 it was shown that given a matrix of communications levels between nodes a corresponding configuration in two dimensions can be found. Suppose one collects the number of communications between nodes for two different one week periods, yielding two weighted adjacency matrices of communications between nodes. Obviously, it is unlikely that the two weighted adjacency matrices will be identical, but is it possible to determine how similar the two matrices are to each other?

The question above is similar to the question asked in a linear regression, how well is the observed value of the response variable explained by the value of the explanatory variable? The two most common ways to measure the fit of a regression line are the coefficient of correlation and the coefficient of determination. In this work, there are four regression correlation coefficients that were considered: 1) procrustes; 2) Euclidean n-dimensional regression coefficient; 3) Affine n-dimensional regression coefficient; and 4) Projective n-dimensional regression coefficient. At the end of this section, two examples illustrating the capabilities of each of the methods discussed will be provided. What immediately follows are the derivations of the respective methods.

2.2.1 Procrustes

Procrustes is a method of comparing two configurations of points in Euclidean space (Source: Cox and Cox, page 92). Procrustes seeks to minimize D^2, the sum of the individual distances between distinct pairs of points. Sibson (1978) formalized the definitions used and the algebra behind procrustes analysis by codifying the work done by Hurley and Cattell (1962) in factor analysis, and Schonemann and Carroll (1970) and Gower (1971) in multi-dimensional scaling; Mardia, et al (1979) proved the conclusions of Sibson by using matrix calculus. The definitions and equations provided in this section are a combination of those used by Cox and Cox based on Sibson's work (1978) as well as the derivative based approach used by Mardia et al. Given two configurations of points, $\boldsymbol{A}$ and $\boldsymbol{B}$, Sibson breaks procrustes down into three main steps:

I. Standardize $\boldsymbol{A}$ and $\boldsymbol{B}$ to place the centroid at the origin.

II. Match ***B*** to ***A*** under an orthogonal transformation.

III. Match the transformed ***B*** to ***A*** under dilation (scaling).

The underlying concept of procrustes is to match two configurations of points as closely as possible to each other by holding one configuration as fixed, and then translating, rotating, reflecting, and / or dilating the second configuration such that the sum of square distances between matched pairs is as small as possible. The starting point for this method is the distance formula between two points:

$$d^2 = (x - x')^2 + (y - y')^2 \quad (2.16)$$

Now suppose there are two configurations, $\mathbf{A} = \left[\underline{\mathbf{a_1}}, \underline{\mathbf{a_2}}, \cdots, \underline{\mathbf{a_n}}\right]^T$ and $\mathbf{B} = \left[\underline{\mathbf{b_1}}, \underline{\mathbf{b_2}}, \cdots, \underline{\mathbf{b}_n}\right]^T$ where each $\mathbf{a_i}$ and $\mathbf{b_i}$ represent the coordinate vectors for the i[th] point in each space. Rewriting (4.1) in terms of the coordinate vectors gives:

$$D^2 = \sum_{i=1}^{n} (\mathbf{a_i} - \mathbf{b_i})^T (\mathbf{a_i} - \mathbf{b_i}) \quad (2.17)$$

If the two configurations match exactly, then the value of D^2 will be zero. Suppose both configurations are based on a multi-dimensional scaling decomposition of two (or even the same) distance matrices; then it is possible to have two identical configurations which differ only in scale, rotation, translation, and or reflection. For the purposes of this derivation, let configuration **B** be scaled, rotated, translated, and reflected to best match the **A** configuration.

$$\mathbf{b'_i} = \rho \mathbf{R^T b_i} + \mathbf{c} \quad (2.18)$$

In equation (2.19), **R** represents the orthogonal matrix which yields the desired rigid rotation and reflection of the points; vector **c** is the rigid translation vector; and ρ is the scale parameter. Now each of the $\mathbf{b'_i}$, are the rotated, reflected, scaled, and translated vectors which minimize (2.18).

$$D^2_{min} = \sum_{i=1}^{n} (\mathbf{a_i} - \rho\mathbf{R^T b_i} - \mathbf{c})^T (\mathbf{a_i} - \rho\mathbf{R^T b_i} - \mathbf{c}) \quad (2.19)$$

Optimal Translation

At this point, the derivation presented moves away from Sibson's work and picks up with the work of Mardia et al. To derive the optimal translation, **c**, expand (2.19) and minimize the resulting equation with respect to **c**.

$$D^2_{min} = \sum_{i=1}^{n} \mathbf{a}_i^T\mathbf{a_i} - \mathbf{a}_i^T\rho\mathbf{R^T b_i} - \mathbf{a}_i^T\mathbf{c} - (\rho\mathbf{R^T b_i})^T\mathbf{a_i} + (\rho\mathbf{R^T b_i})^T\rho\mathbf{R^T b_i} - (\rho\mathbf{R^T b_i})^T\mathbf{c}$$

$$-\mathbf{c^T a_i} + \mathbf{c^T}\rho\mathbf{R^T b_i} + \mathbf{c^T c}$$

Collecting like terms gives:

$$D^2_{min} = n\mathbf{c^T c} + \sum_{i=1}^{n} \mathbf{a}_i^T\mathbf{a_i} - 2\mathbf{a}_i^T\rho\mathbf{R^T b_i} + (\rho\mathbf{R^T b_i})^T\rho\mathbf{R^T b_i} - 2\mathbf{c^T a_i} + 2\mathbf{c^T}\rho\mathbf{R^T b_i} \quad (2.20)$$

Taking the partial with respect to $\mathbf{c^T}$ gives:

$$\frac{\partial D^2}{\partial \mathbf{c^T}} = 2n\mathbf{c} + \sum_{i=1}^{n} -2\mathbf{a_i} + 2\rho\mathbf{R^T b_i} \quad (2.21)$$

Setting (2.21) equal to zero and solving for **c** gives**:**

$$n\mathbf{c} = \sum_{i=1}^{n} \mathbf{a_i} - \rho\mathbf{R^T}\mathbf{b_i} \rightarrow \mathbf{c} = \frac{1}{\mathrm{n}}\sum_{i=1}^{n} \mathbf{a_i} - \rho\mathbf{R^T}\frac{1}{\mathrm{n}}\sum_{i=1}^{n} \mathbf{b_i} = \bar{\mathbf{a}} - \rho\mathbf{R^T}\bar{\mathbf{b}} \qquad (2.22).$$

In other words, the optimal translation is achieved when the centroids of both configurations lie at the origin. Since centering the configurations on the origin is very simple and can be done prior to rotating and / or dilating one of the configurations to match the other, the derivation of the optimal rotation and optimal dilation will assume that **A** and **B** are centered at the origin.

Optimal Rotation

Assume the two configurations, **A** and **B,** are both centered at the origin. Hold **A's** configuration fixed; rotate and dilate **B's** configuration to match **A's** as closely as possible. Now, equation (2.17) can be written as:

$$D^2 = \sum_{i=1}^{n} (\mathbf{a_i} - \rho\mathbf{R^T}\mathbf{b_i})^T(\mathbf{a_i} - \rho\mathbf{R^T}\mathbf{b_i}) \qquad (2.23)$$

Here **R** is the rotation matrix for which $\mathbf{b_i''} = \mathbf{R^T}\mathbf{b_i}$ matches, as closely as possible, the orientation of $\mathbf{a_i}$, and $\mathbf{b_i'} = \rho\mathbf{R^T}\mathbf{b_i}$ gives the best match to $\mathbf{a_i}$ in both orientation and scale. A desirable property, and a restriction that will be used in the following derivation, is for **R** to be an orthogonal matrix.

Expanding (2.23) gives:

$$D^2 = \sum_{i=1}^{n} \mathbf{a}_i^T \mathbf{a_i} - \mathbf{a}_i^T \rho \mathbf{R^T b_i} - (\rho \mathbf{R^T b_i})^T \mathbf{a_i} + (\rho \mathbf{R^T b_i})^T \rho \mathbf{R^T b_i}$$

$$= \sum_{i=1}^{n} \mathbf{a}_i^T \mathbf{a_i} - 2\mathbf{a}_i^T \rho \mathbf{R^T b_i} + \rho^2 \mathbf{b_i^T R\, R^T b_i} \qquad (2.24)$$

Notice that if $\boldsymbol{R}$ is orthogonal, equation (2.24) reduces to:

$$= \sum_{i=1}^{n} \mathbf{a}_i^T \mathbf{a_i} - 2\rho \mathbf{a}_i^T \mathbf{R^T b_i} + \rho^2 \mathbf{b_i^T b_i}$$

$$= tr(\boldsymbol{AA^T}) - 2\rho\{trace(\boldsymbol{AR^T B^T})\} + \rho^2\{trace(\boldsymbol{BB^T})\} \quad (2.25)$$

Here equation (2.25) is in the general form of the procrustes statistic as defined by Sibson (1978). By constraining $\boldsymbol{R}$ to being orthogonal, the minimization of (2.25) is equivalent to maximizing the $\{trace(\boldsymbol{AR^T B^T})\}$. By definition, $\boldsymbol{R^T R = RR^T = I}$; since for any matrix, $\boldsymbol{M}$, the matrix product $\boldsymbol{M^T M}$ is symmetric, and $\boldsymbol{r_i^T r_j} = 1$ when $i = j$ and $\boldsymbol{r_i^T r_j} = 0$ when $i \neq j$. This gives rise to a total of $k(k+1)/2$ constraints. Let $\boldsymbol{\Lambda}$ be a k x k matrix of Lagrangian multipliers; maximizing $\{trace(\boldsymbol{AR^T B^T})\}$ with respect to $\boldsymbol{R}$ and subject to the constraint that $\boldsymbol{R}$ be orthogonal is equivalent to maximizing:

$$trace\left(\boldsymbol{AR^T B^T} - \frac{1}{2}\boldsymbol{\Lambda}(\boldsymbol{RR^T - I})\right) = trace(\boldsymbol{A^T BR}) - \frac{1}{2}trace(\boldsymbol{\Lambda RR^T}) + \frac{1}{2}trace(\boldsymbol{\Lambda})$$

$$= trace(\boldsymbol{R^T B^T A}) - \frac{1}{2}trace(\boldsymbol{R^T \Lambda R}) + \frac{1}{2}trace(\boldsymbol{\Lambda}) \quad (2.25a)$$

The partial derivative of (2.25a) with respect to $\boldsymbol{R^T}$ is:

$$\boldsymbol{B^T A - \Lambda R} \quad (2.25b)$$

Let $\boldsymbol{Z = B^T A}$. Setting (2.25b) equal to zero and post multiplying by $\boldsymbol{R^T}$ gives:

$$\boldsymbol{Z = \Lambda R} \quad (2.25c)$$

$$\boldsymbol{\Lambda = ZR^T \rightarrow \Lambda^2 = ZR^T RZ^T = ZZ^T} \quad (2.25d)$$

Note that **Z** is a real matrix with *rank* ($\boldsymbol{Z}$) ≥ 0. By the singular value decomposition theorem, $\boldsymbol{Z}$ can be represented as the product of three matrices, $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{\Gamma}$ as follows:

$$\boldsymbol{Z = V\Gamma U^T} \quad (2.25e)$$

Here, $\boldsymbol{U}$ and $\boldsymbol{V}$ are *k x k* orthogonal matrices, and $\boldsymbol{\Gamma}$ is a *k x k* diagonal matrix with all positive diagonal elements. Substituting (2.25e) into (2.25d) gives:

$$\boldsymbol{\Lambda^2 = V\Gamma U^T U\Gamma V^T = V\Gamma\Gamma V^T = V\Gamma V^T V\Gamma V^T}$$

$$\boldsymbol{\Lambda = V\Gamma V^T} \quad (2.25f)$$

Now, substituting (2.25f) into (2.25c) and combining with (2.25e) gives:

$$\boldsymbol{V\Gamma U^T = V\Gamma V^T R}$$

$$\boldsymbol{\widehat{R} = (V\Gamma V^T)^{-1} V\Gamma U^T = V\Gamma^{-1} V^T V\Gamma U^T = V\Gamma^{-1}\Gamma U^T = VU^T} \quad (2.25g)$$

This gives an optimal rotation matrix, $\boldsymbol{\widehat{R}}$, which is orthogonal since both $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal.

Recall that $\boldsymbol{Z = B^T A = V\Gamma U^T}$ and from equation (2.25d):

$$\boldsymbol{\Lambda^2 = ZR^T RZ^T = ZZ^T} \quad (2.25d)$$

$$\boldsymbol{ZZ^T = V\Gamma U^T U\Gamma V^T = V\Gamma V^T V\Gamma V^T}$$

$$\boldsymbol{\Lambda = V\Gamma V^T} \quad (2.25h)$$

Now, substitute (2.25h) into $(2.25c)$; combine with $(2.25e)$ and solve for $\boldsymbol{R}$:

$$\boldsymbol{Z = V\Gamma V^T R}$$

$$\boldsymbol{\widehat{R} = (V\Gamma V^T)^{-1} Z = V\Gamma^{-1} V^T V\Gamma U^T = VU^T} \quad (2.26)$$

Using the property of trace of a matrix, the second term of (2.25) can be rewritten as:

$$D^2 = tr(\boldsymbol{AA^T}) - 2\rho\{trace(\boldsymbol{BRA^T})\} + \rho^2\{trace(\boldsymbol{BB^T})\}$$

Substituting $\boldsymbol{Z}$ for $\boldsymbol{B^T A}$ gives:

$$D^2 = tr(\boldsymbol{AA^T}) - 2\rho\{trace(\boldsymbol{Z^T R})\} + \rho^2\{trace(\boldsymbol{BB^T})\}$$

Finally, substituting $\boldsymbol{VU^T}$ for $\boldsymbol{R}$; and $\boldsymbol{V\Gamma U^T}$ for $\boldsymbol{Z}$ gives**:**

$$D^2 = tr(\boldsymbol{AA^T}) - 2\rho\{trace(\boldsymbol{U\Gamma V^T VU^T})\} + \rho^2\{trace(\boldsymbol{BB^T})\}$$

$$D^2 = tr(\boldsymbol{AA^T}) - 2\rho\{trace(\boldsymbol{\Gamma})\} + \rho^2\{trace(\boldsymbol{BB^T})\} \quad (2.27)$$

Notice that the second term in equation (2.27) is equivalent to:

$$2\rho\{trace(\boldsymbol{V^T V\Gamma})\} = 2\rho\{trace(\boldsymbol{V\Gamma V^T})\}$$

Which, from equation (2.25d), is equal to:

$$2\rho\left\{trace(\boldsymbol{\Lambda^2})^{1/2}\right\} = 2\rho\left\{trace(\boldsymbol{ZZ^T})^{1/2}\right\} = 2\rho\left\{trace(\boldsymbol{Z^TZ})^{1/2}\right\}$$

$$= 2\rho\left\{trace(\boldsymbol{A^TBB^TA})^{1/2}\right\} \quad (2.28)$$

So the final form of the general procrustes statistic is:

$$D^2 = tr(\boldsymbol{AA^T}) + \rho^2\{trace(\boldsymbol{BB^T})\} - 2\rho\left\{trace(\boldsymbol{A^TBB^TA})^{1/2}\right\} \quad (2.29)$$

To solve for $\widehat{\boldsymbol{R}}$, the argument of Sibson will be employed. Recall that minimizing equation 2.25 is the same as maximizing the $trace(\boldsymbol{AR^TB^T})$.

Sibson starts by noting that the implication of the derivation of 2.29 starting with equation 2.25 is that:

$$trace(\boldsymbol{AR^TB^T}) = trace(\boldsymbol{R^TB^TA}) = trace(\boldsymbol{R^TZ})$$

$$= trace(\boldsymbol{R^TV\Gamma U^T}) = trace(\boldsymbol{U^TR^TV\Gamma}) \quad (2.30)$$

The matrix $\boldsymbol{U^TR^TV}$ is orthogonal since it is a product of orthogonal matrices.

Proof: Let $\boldsymbol{U}$, $\boldsymbol{R}$, $\boldsymbol{V}$ be orthogonal matrices, and let $\boldsymbol{P} = \boldsymbol{U^TR^TV}$, then

$$\boldsymbol{P^TP} = (\boldsymbol{U^TR^TV})^T\boldsymbol{U^TR^TV} = \boldsymbol{V^TRUU^TR^TV} = \boldsymbol{V^TRR^TV} = \boldsymbol{V^TV} = \boldsymbol{I} \text{ and,}$$

$$\boldsymbol{PP^T} = \boldsymbol{U^TR^TV}(\boldsymbol{U^TR^TV})^T = \boldsymbol{U^TR^TV^TRU} = \boldsymbol{U^TR^TRU} = \boldsymbol{U^TU} = \boldsymbol{I}$$

Thus, $\boldsymbol{P}$ is an orthogonal matrix**.**

Since $\boldsymbol{U^TR^TV}$ is orthogonal, every element is bounded between -1 and 1; this coupled with equation 2.30 implies that:

$$trace\left(\boldsymbol{U^T R^T V \Gamma}\right) \leq trace(\boldsymbol{\Gamma}) = trace(\boldsymbol{\Gamma}^T \boldsymbol{\Gamma})^{0.5}$$

With equality only when $\boldsymbol{R^T = UV^T}$. This result is consistent with Mardia's derivation, only now Sibson introduces the condition that $\boldsymbol{U^T R^T V \Gamma = \Gamma}$. This implies that

$$\boldsymbol{R^T V \Gamma U^T = U \Gamma U^T \rightarrow R^T Z = U \Gamma U^T = [U \Gamma U^T U \Gamma U^T]^{1/2}}$$

$$\boldsymbol{= [U \Gamma^2 U^T]^{1/2} = [U \Gamma V^T V \Gamma U^T]^{1/2} = [Z^T Z]^{1/2}}$$

$$\boldsymbol{R^T Z = [Z^T Z]^{1/2}} \quad (2.31)$$

$$\widehat{\boldsymbol{R^T}} = \boldsymbol{(A^T B B^T A)^{1/2} (B^T A)^{-1}} \quad (2.32)$$

Optimal Dilation

To derive the optimal dilation, ρ, equation (2.25) is minimized with respect to ρ. The partial derivative of D^2 with respect to ρ is:

$$\frac{\partial D^2}{\partial \rho} = -2\{trace(\boldsymbol{ARB^T})\} + 2\rho\{trace(\boldsymbol{BB^T})\}$$

Setting the partial derivative equal to zero and solving for ρ gives:

$$\hat{\rho} = \frac{trace(\boldsymbol{ARB^T})}{trace(\boldsymbol{BB^T})} \quad (2.33)$$

However, under the condition that $\boldsymbol{U^T R^T V \Gamma = \Gamma}$, and equations 2.31, 2.32, and 2.33 the final computational form of the optimal dilation is:

$$\hat{\rho} = \frac{trace(\boldsymbol{A^T B B^T A})^{1/2}}{trace(\boldsymbol{B B^T})} \qquad (2.34)$$

Procrustes is a configuration matching method that accounts for translation, by centering both configurations on the origin; followed by calculation of the optimal rotation / reflection and optimal dilation (scale). There are two readily identifiable limitations to procrustes: 1) the method only works when all matrix products are positive semi-definite; and 2) with only a single overall scaling parameter; different dilations in each principle direction are "averaged" into one overall best fit scale estimate.

2.2.2 Bidimensional Regression

In his 1994 paper "Bidimensional Regression", Tobler defined four methods of comparing the degree of resemblance between two planar figures. Three of the methods are geometrically linear in that "straight lines in the original are also straight lines in the image". The fourth method is curvilinear and will not be discussed in this work.

Tobler derives the four bidimensional transformations by analogy with the ordinary least squares univariate linear model. Here, the term linear does not mean the method only models lines, but rather the model is a polynomial that is linear in its coefficients. The idea behind a linear model is to look at the relationship or association between some response variable, y_i, and a set of explanatory variables, $x_1, x_2, \cdots, x_p$ as in equation 2.35.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + e_i \quad (2.35)$$

where $e_i \widetilde{iid} N(0,\ \sigma^2)$. Now suppose there are n observations total; giving rise to n linear models:

$$y_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_p x_{1p} + e_1$$

$$y_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_p x_{2p} + e_2$$

$$\vdots$$

$$y_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_p x_{np} + e_n$$

Then the linear model for all n observations can be written in matrix notation as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} \quad (2.36)$$

Or simply as $\boldsymbol{Y} = \boldsymbol{XB} + \boldsymbol{E}$. Taking the expected values of both sides shows that the expected value of $\boldsymbol{Y}$ is just $\boldsymbol{XB}$. So now the question is, how well does the expected value of $\boldsymbol{Y}$ (E($\boldsymbol{Y}$) = $\boldsymbol{XB}$) from the model, fit the observed values $\boldsymbol{Y}$ compared to the arithmetic mean of $\boldsymbol{Y}$. One of the most well known measures of linear regression fit is the coefficient of determination (R^2). The coefficient of determination is the ratio of the difference in sums of squares from the mean and sums of squares for the model to the sums of squares from the mean.

$$R^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2 - \sum_{i=1}^{n}(y_i - \hat{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (2.37)$$

In other words, 100 times R^2 is the percent variation in the observations explained by the model over that which is explained by fitting only the arithmetic mean. Analogous to the linear model, Tobler defines three geometrically linear transformations in the context of linear models, and defines a bidimensional coefficient of determination as a measure of how well two configurations match each other.

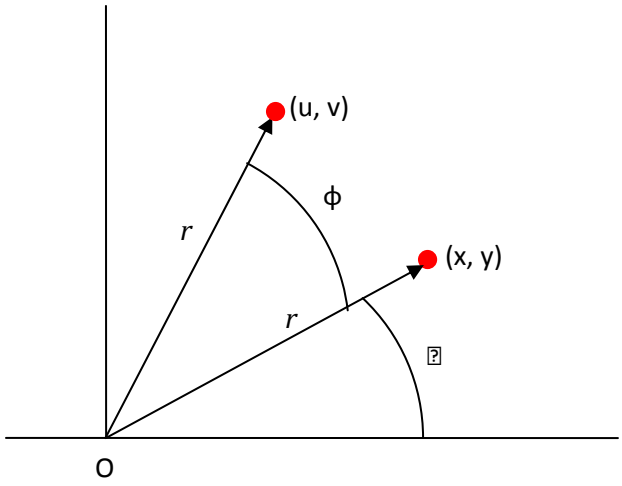
2.2.2.1 Euclidean Transformation

Tobler named his first method the Euclidean transformation. Although originally derived using complex number theory [Tobler, 1994], this method is easily derived using the relationship between cylindrical and Cartesian coordinates along with an application of Euler's identity. The Euclidean transformation is similar to the procrustes method in that it accounts for translation, rotation / reflection, and a single overall dilation (scale).

Consider the situation depicted in figure 2-11. Here, the point (*x, y*) represents the original configuration; the point (*u,v*) represents a second configuration of the same object, perhaps just measured on a different scale, or from a different reference point. From geometry, in two dimensions, any real Cartesian coordinate can be represented as a cylindrical coordinate as follows:

$$x = r\cos(\theta); \text{and } y = r\sin(\theta) \quad (2.38)$$

Figure 2-19: Rotation of a point about the origin.



Where *r* is the distance the point (*x, y*) lies from the origin; and is the angle between the *x* – axis and *r* measured in the counter-clockwise direction if is positive, and the clockwise direction if is negative. Now consider rotating the point (*x, y*) an additional φ degrees in the counter –clockwise direction to the new point (*u, v*). Notice that the distance (*u, v*) lies from the origin is r. Therefore, representing (*u, v*) in cylindrical coordinates and applying Euler's identity:

$$u = r\cos(\theta + \phi) \ = r[\cos(\theta)\cos(\phi) - \sin(\theta)\sin(\phi)] \quad (2.39a)$$

$$v = r\sin(\theta + \phi) = r[\cos(\theta)\sin(\phi) + \ \sin(\theta)\cos(\phi)] \quad (2.39b)$$

Distributing the *r*, and substituting (2.38) into (2.39a and 2.39b) gives:

$$u = x\cos(\phi) - y\sin(\phi) \ \ (2.40a)$$

$$v = x\sin(\phi) + y\cos(\phi) \ \ (2.40b)$$

Now, allowing for different translation distances (α_1, α_2) in each direction, and a uniform scaling factor (μ), (2.40a) and (2.40b) become:

$$u = \alpha_1 + \mu\{x\cos(\phi) - y\sin(\phi)\} \quad (2.41a)$$

$$v = \alpha_2 + \mu\{x\sin(\phi) + y\cos(\phi)\} \quad (2.41b)$$

Equations 2.41a and 2.41b can be combined into a matrix form:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \mu \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.42)$$

Tobler (1994) linearizes 2.42 as follows to yield the Euclidean transformation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} \beta_1 & -\beta_2 \\ \beta_2 & \beta_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.43)$$

Where:

$$\phi = tan^{-1}\left(\frac{\beta_2}{\beta_1}\right) \quad (2.44a) \text{ and } \mu = \sqrt{(\beta_1^2 + \beta_2^2)} \quad (2.44b)$$

It is clear from the derivation that Euclidean Bidimensional regression will perform at its best, when any two configurations differ only in rotation and overall scale.

In practice, the α_i and the β_i in equation 2.43 are unknown parameters and therefore must be estimated. Parameter estimates can be found by placing the α_i and β_i in a column vector and rewriting equation 2.43 in the familiar linear model form (i.e. ***W=ZB***):

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \\ v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_1 & -y_1 \\ 1 & 0 & x_2 & -y_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_n & -y_n \\ 0 & 1 & y_1 & x_1 \\ 0 & 1 & y_2 & x_2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & y_n & x_n \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

Where:

$$\boldsymbol{W} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \\ v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}; \boldsymbol{Z} = \begin{bmatrix} 1 & 0 & x_1 & -y_1 \\ 1 & 0 & x_2 & -y_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_n & -y_n \\ 0 & 1 & y_1 & x_1 \\ 0 & 1 & y_2 & x_2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & y_n & x_n \end{bmatrix}; \text{and } \boldsymbol{B} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

By utilizing a linear model, Tobler is implicitly assuming the following:

$$cov(u_i, u_j) = cov(v_i, v_j) = cov(u_i, v_j) = 0 \text{ when } i \neq j, and$$

$$cov(u_i, u_j) = cov(v_i, v_j) = \sigma^2, \text{and } cov(u_i, v_j) = 0 \text{ when } i = j$$

Using these assumptions, the least squares estimate of $\boldsymbol{B}$ is:

$$\widehat{\boldsymbol{B}} = (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{W}$$

$$\boldsymbol{Z}'\boldsymbol{Z} = \begin{bmatrix} n & 0 & \sum_i x_i & -\sum_i y_i \\ 0 & n & \sum_i y_i & \sum_i x_i \\ \sum_i x_i & \sum_i y_i & \sum_i (x_i^2 + y_i^2) & 0 \\ -\sum_i y_i & \sum_i x_i & 0 & \sum_i (x_i^2 + y_i^2) \end{bmatrix} \quad (2.45)$$

The expected values of the coordinates using the Euclidean transformation are:

$$\boldsymbol{Z\hat{B}} = \boldsymbol{Z(Z'Z)^{-1}Z'W}$$

And the Euclidean coefficient of determination is found as follows:

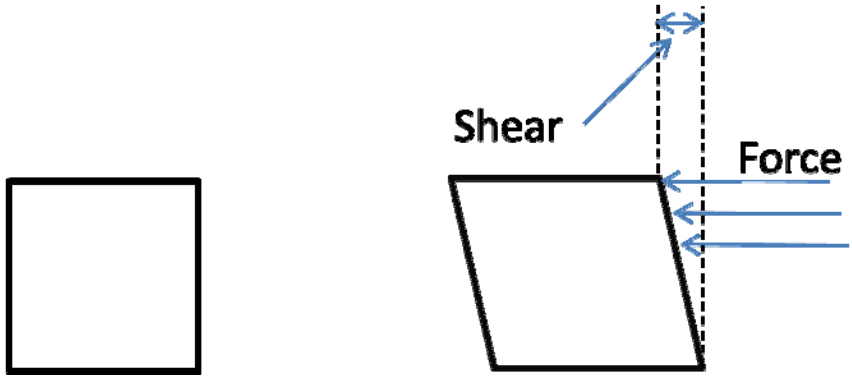
$$R^2_{Euclidean} = 1 - \frac{\left(\boldsymbol{W - Z\hat{B}}\right)'\left(\boldsymbol{W - Z\hat{B}}\right)}{(\boldsymbol{W - \overline{W}})'(\boldsymbol{W - \overline{W}})} \quad (2.46)$$

Just as with the univariate, one dimensional coefficient of determination, the Euclidean coefficient of determination can only take on values between 0 and 1; where a value near 0 indicates a poor fit (i.e. the configurations are different) and a value near 1 indicates a good fit (i.e. the configurations are similar).

2.2.2.2 Affine Transformation

Tobler's second configuration matching technique is known as the affine transformation, or affine bidimensional regression. This technique was developed to account for different amounts of scaling in each cardinal direction, rigid rotation, reflection, as well as possible shear. Shear is the distortion of an object in a coplanar direction. For example, consider a square whose bottom is fixed; whose sides are rigid and whose vertices are flexible. Now, suppose a force, acting from right to left, is applied to the upper right corner of the square. The entire top edge of the square will move down and to the left due to the force. The length of movement of the square's corner from its original position is called shear.

Figure 2-20: Example of Shear.



The affine transformation is defined in equation 2.47 [Tobler, 1994]

Or

As with the Euclidean transformation, Tobler places both the new and the original configuration coordinate pairs into column vectors with the *x* and *u* coordinates stacked atop the *y* and *v* coordinates, preserving the pair order so that equation (2.47) can be analyzed as a univariate linear model.

Tobler's original parameterization requires the design matrix (Z) to be set up as follows:

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \\ v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_1 & y_1 & 0 & 0 \\ 1 & 0 & x_2 & y_2 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & x_n & y_n & 0 & 0 \\ 0 & 1 & 0 & 0 & x_1 & y_1 \\ 0 & 1 & 0 & 0 & x_2 & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & x_n & y_n \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}$$

Solving the normal equations for the linear model in (2.47) gives:

$$\hat{B} = \begin{bmatrix} n & 0 & \sum_i x_i & \sum_i y_i & 0 & 0 \\ 0 & n & 0 & 0 & \sum_i x_i & \sum_i y_i \\ \sum_i x_i & 0 & \sum_i x_i^2 & \sum_i x_i y_i & 0 & 0 \\ \sum_i y_i & 0 & \sum_i x_i y_i & \sum_i y_i^2 & 0 & 0 \\ 0 & \sum_i x_i & 0 & 0 & \sum_i x_i^2 & \sum_i x_i y_i \\ 0 & \sum_i y_i & 0 & 0 & \sum_i x_i y_i & \sum_i y_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_i u_i \\ \sum_i v_i \\ \sum_i x_i u_i \\ \sum_i y_i u_i \\ \sum_i x_i v_i \\ \sum_i y_i v_i \end{bmatrix} \quad (2.48)$$

However, shifting the order of the parameters in the ***B*** matrix, such that the first three parameters are the intercept, the x-coefficient, and the y-coefficient for the u_i and the last three parameters are the intercept; the x-coefficient, and the y-coefficient for the v_i the model becomes:

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \\ v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ 1 & x_2 & y_2 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & y_n & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ 0 & 0 & 0 & 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & x_n & y_n \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \beta_1 \\ \beta_2 \\ \alpha_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} \quad (2.49)$$

Notice that the design matrix in (2.49) is a block diagonal where each block is identical. This means that the affine transformation parameters can be estimated by multivariate, multiple regression.

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ \vdots & \vdots \\ u_n & v_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_3 \\ \beta_2 & \beta_4 \end{bmatrix} (2.50)$$

Let,

$$\boldsymbol{Y} = \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ \vdots & \vdots \\ u_n & v_n \end{bmatrix} \text{ and } y_{i1} = u_i;\ y_{i2} = v_i$$

then when $\boldsymbol{Y}$ is rearranged into a column vector, Tobler implicitly assumes that each row of $\boldsymbol{Y}$ is mutually independent of all other rows in $\boldsymbol{Y}$. Further it is implicitly assumed that each u_i is independent of all v_j, where i and j can but are not necessarily equal. These assumptions are consistent with those made in multivariate, multiple regression, see Johnson and Wichern (2007). The solution to a multivariate, multiple regression is:

$$\widehat{\boldsymbol{B}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

Where $\boldsymbol{X}$ is the design matrix and $\boldsymbol{Y}$ is the *n x 2* matrix of coordinates for the new configuration. The solution to equation (2.50) is:

$$\widehat{\boldsymbol{B}} = \begin{bmatrix} n & \sum_i x_i & \sum_i y_i \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i y_i \\ \sum_i y_i & \sum_i x_i y_i & \sum_i y_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_i u_i & \sum_i v_i \\ \sum_i x_i u_i & \sum_i x_i v_i \\ \sum_i y_i u_i & \sum_i y_i v_i \end{bmatrix} \quad (2.51)$$

Using the estimated coefficients from equation 2.51, one can solve for the expected coordinates for the new configuration, $\widehat{\boldsymbol{Y}}$, as follows:

$$\hat{Y} = \boldsymbol{X}\widehat{\boldsymbol{B}} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

The affine coefficient of determination is defined in equation (2.52) and is interpreted in the same manner as the Euclidean coefficient of determination.

$$R^2_{Affine} = 1 - \frac{\left(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{B}}\right)'\left(\boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{B}}\right)}{(\boldsymbol{Y} - \overline{\boldsymbol{Y}})'(\boldsymbol{Y} - \overline{\boldsymbol{Y}})} \quad (2.52)$$

One of the drawbacks to the affine transformation is that it can be difficult to interpret the meaning of the coefficients. Consider two situations:

1) Case 1: The two configurations differ in rotation, and have a different scaling factor in each cardinal direction.
2) Case 2: The two configurations differ in rotation; have a different scaling factor in each cardinal direction; and there is some amount of shear.

In case 1, the interpretation of the coefficients is fairly straight forward since the model reduces to a hybrid Euclidean model (equation 2.43).

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} \beta_1 & \beta_2 \\ \beta_3 & \beta_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} \mu\cos(\phi) & -\mu\sin(\phi) \\ \rho\sin(\phi) & \rho\cos(\phi) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.53)$$

In equation 2.53, $\boldsymbol{\mu}$ is the scale factor in the x direction and $\boldsymbol{\rho}$ is the scaling factor in the y direction and are found as follows:

$$\mu = \sqrt{\beta_1^2 + \beta_2^2} = \sqrt{\left(\mu\cos(\phi)\right)^2 + \left(-\mu\sin(\phi)\right)^2} \quad (2.54a)$$

$$\rho = \sqrt{\beta_3^2 + \beta_4^2} = \sqrt{\left(\rho\sin(\phi)\right)^2 + \left(\rho\cos(\phi)\right)^2} \quad (2.54b)$$

The angle of rotation,ϕ, can be found by taking the arc-tangent of the ratio of $-\beta_2$ to β_1 or β_3 to β_4.

$$\phi = \tan^{-1}\left(-\frac{\beta_2}{\beta_1}\right) \quad (2.55a)$$

$$\phi = \tan^{-1}\left(\frac{\beta_3}{\beta_4}\right) \quad (2.55b)$$

It is important to note, that equations (2.55a and b) only hold if there is no shear present.

In case 2, the introduction of shear makes it impossible to separate out the elements of $\boldsymbol{B}$ which can be combined to give the scale and the angle of rotation. The reason behind this is that under shear, certain points will lie where expected under a Euclidean rigid body rotation, but the points that were sheared, will appear to be either under-rotated, or over-rotated when compared to the rigid rotation angle. The easiest way to detect the presence of shear is to solve for the rotation angle using both equation 2.55a and 2.55b. If the two angles obtained are not equal, shear is present.

2.2.2.3 Projective Transformation

The last of Tobler's configuration matching techniques that were considered is the projective transformation. This is a geometrically linear transformation (straight lines in the original configuration are straight lines in the new configuration, but is nonlinear in its parameters. The non-linearity is due to the introduction of an overall weighting vector, ***T***. Here, each element of ***T*** is a weight applied to its associated new configuration coordinate. Essentially, the projective transformation allows for a different scaling factor being applied to the individual coordinate elements. The vector of weights is also modeled as a function of the original configuration. The projective transformation is defined as follows:

$$u = \frac{\beta_1 + \beta_2 x + \beta_3 y}{\beta_7 + \beta_8 x + \beta_9 y} \quad (2.56a)$$

$$v = \frac{\beta_4 + \beta_5 x + \beta_6 y}{\beta_7 + \beta_8 x + \beta_9 y} \quad (2.56b)$$

Where $\boldsymbol{T} = \beta_7 + \beta_8 x + \beta_9 y$. Putting this into Tobler's vectorized linear model format requires some arithmetic manipulation by substituting T into equations 2.56a and b and multiplying both sides by T, the equations can be rewritten as:

$$Tu = \beta_1 + \beta_2 x + \beta_3 y \quad (2.57a)$$

$$Tv = \beta_4 + \beta_5 x + \beta_6 y \quad (2.57b)$$

$$T = \beta_7 + \beta_8 x + \beta_9 y \quad (2.57c)$$

Now, equations 2.57a-c can be put into a vector format:

$$\begin{bmatrix} Tu \\ Tv \\ T \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_4 \\ \beta_7 \end{bmatrix} + \begin{bmatrix} \beta_2 & \beta_3 \\ \beta_5 & \beta_6 \\ \beta_8 & \beta_9 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Just as with the affine transformation, this can be rewritten into a multivariate, multiple regression model:

$$\begin{bmatrix} t_1u_1 & t_1v_1 & t_1 \\ t_2u_2 & t_2v_2 & t_2 \\ \vdots & \vdots & \vdots \\ t_nu_n & t_nv_n & t_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix} \begin{bmatrix} \beta_1 & \beta_4 & \beta_7 \\ \beta_2 & \beta_5 & \beta_8 \\ \beta_3 & \beta_6 & \beta_9 \end{bmatrix}$$

Notice here that the design matrix for the projective transformation is the same as that for the affine transformation.

The solution to this system of equations is:

$$\widehat{\boldsymbol{B}} = \begin{bmatrix} n & \sum_i x_i & \sum_i y_i \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i y_i \\ \sum_i y_i & \sum_i x_i y_i & \sum_i y_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_i t_i u_i & \sum_i t_i v_i & \sum_i t_i \\ \sum_i x_i t_i u_i & \sum_i x_i t_i v_i & \sum_i x_i t_i \\ \sum_i y_i t_i u_i & \sum_i y_i t_i v_i & \sum_i y_i t_i \end{bmatrix}$$

Now, consider the situation where every element of $\boldsymbol{T}$ is the same constant, say $t_i = c$; the equations 2.57a-c reduce to the equations in 2.58a-c; and it is apparent that $\beta_7 = c, \beta_8 = 0,$ and $\beta_9 = 0$.

$$cu = \beta_1 + \beta_2 x + \beta_3 y \quad (2.58a)$$

$$cv = \beta_4 + \beta_5 x + \beta_6 y \quad (2.58b)$$

$$c = \beta_7 + \beta_8 x + \beta_9 y \quad (2.58c)$$

Equations 2.58a-c now reduce to:

$$u = \frac{\beta_1}{c} + \frac{\beta_2}{c} x + \frac{\beta_3}{c} y = \beta_1^* + \beta_2^* x + \beta_3^* y$$

$$v = \frac{\beta_4}{c} + \frac{\beta_5}{c} x + \frac{\beta_6}{c} y = \beta_4^* + \beta_5^* x + \beta_6^* y$$

$$c = \beta_7$$

Which is the affine transformation; therefore whenever every element of T is the same constant, the projective transformation reduces to the affine transformation. More succinctly, the affine transformation is a special case of the projective transformation.

Although the projective transformation is theoretically very flexible in matching configurations, actual implementation comes with some problems. First, the use of a nonlinear estimation program is required to get a vector of weights from which to start. Second, the use of a weighting vector can actually result in two configurations being rated as a "strong" match even though the same coordinate may have changed relative positions from one configuration to the other. For this reason, the projective transformation will not be considered for further evaluation.

2.3 Examples of Configuration Matching in Two Dimensions

Separate R functions were written that implement each of the following configuration matching techniques: procrustes, Euclidean, affine, and projective; and can be found in Appendix A. Although not being considered further, the projective function is included for completeness. Consider a simple square with the following initial configuration (1, 1), (-1, 1), (-1, -1), and (1, -1). This initial configuration will be progressively perturbed to test each of the techniques' ability to match configurations. The perturbations will be as follows:

1) The centroid of the original configuration translated from (0, 0) to (1, -1).
2) The configuration in 1) rotated 45 degrees counter-clockwise.
3) The configuration in 2) scaled 1.5 times in both x and y.
4) The configuration in 2) scaled 2 times in x and 1.5 times in y.
5) The configuration in 4) sheared, with segment 12 sheared to the left relative to segment 34.

The original configuration is plotted in figure 2-13 and the perturbations described in 1) through 5) above are plotted in figures 2-14 through 2-18. Following each plot is a table summarizing the coefficient of determination, the x and y translation, the rotation angle, the scale, and the presence of shear as determined by each of the configuration matching techniques.

Figure 2-21: Original Configuration.

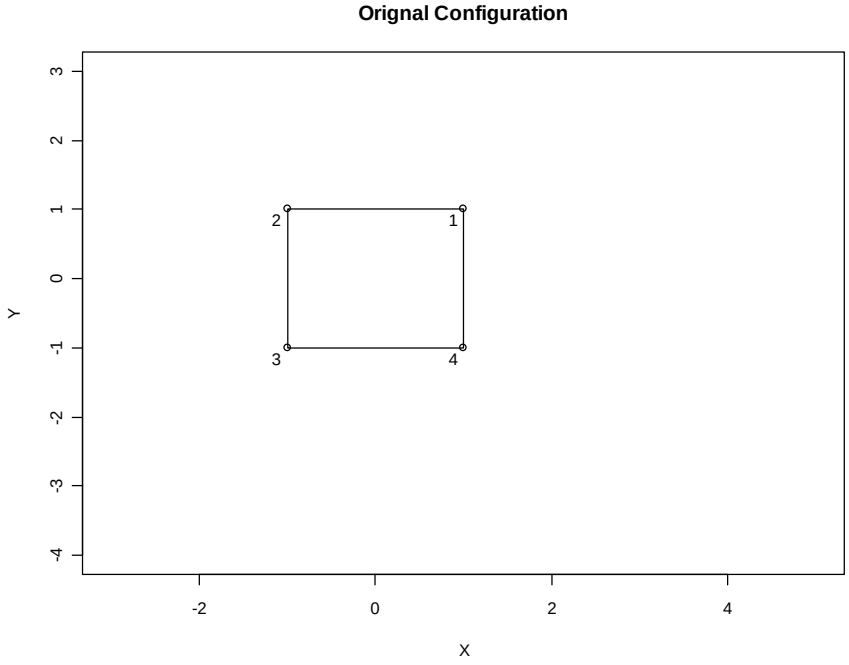


Figure 2-22: Centroid of Original Configuration Translated to (1, -1).

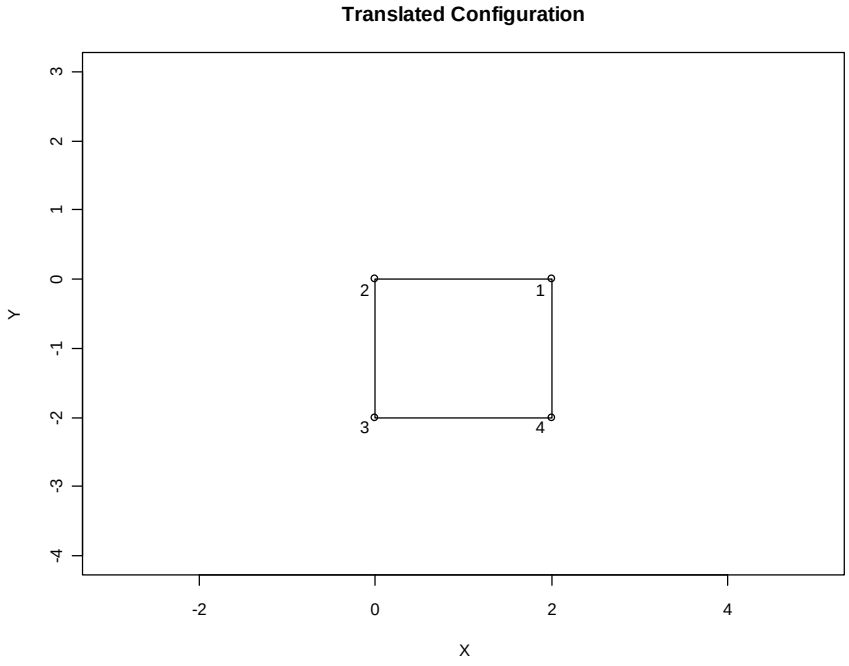
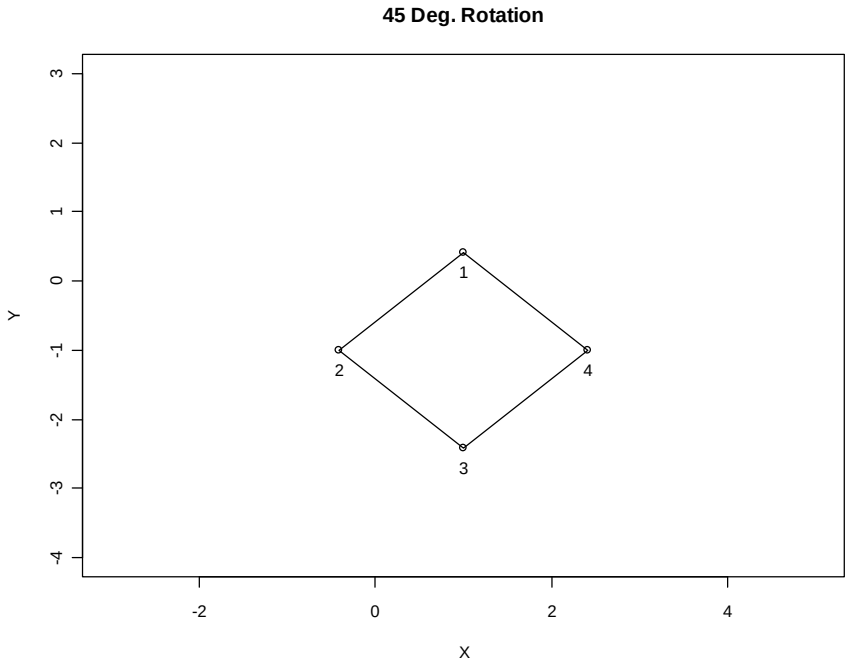


Table2-5: Fit Results for Translated Configuration.

	Procrustes	Euclidean	Affine
R^2	1	1	1
Translation in x	NA[3]	1	1
Translation in y	NA	-1	-1
Scale in x	1[4]	1[5]	1
Scale in y	NA	NA	1
Rotation Angle	0	0	0
Shear Present	Unknown	Unknown	No

Figure 2-23: Translated and Rotated Configuration.



[3] Procrustes centers both configurations at (0, 0) and does not "calculate" translation.
[4] Procrustes assumes only one overall scaling factor; reported here under Scale in x.
[5] Euclidean assumes only one overall scaling factor; reported here under Scale in x.

Table 2-6: Fit Results for Translated and Rotated Configuration.

	Procrustes	Euclidean	Affine
R^2	1	1	1
Translation in x	NA	1	1
Translation in y	NA	-1	-1
Scale in x	1	1	1
Scale in y	NA	NA	1
Rotation Angle	45	45	45
Shear Present	Unknown	Unknown	No

Figure 2-24: Translated, Rotated, and Equally Scaled Configuration.

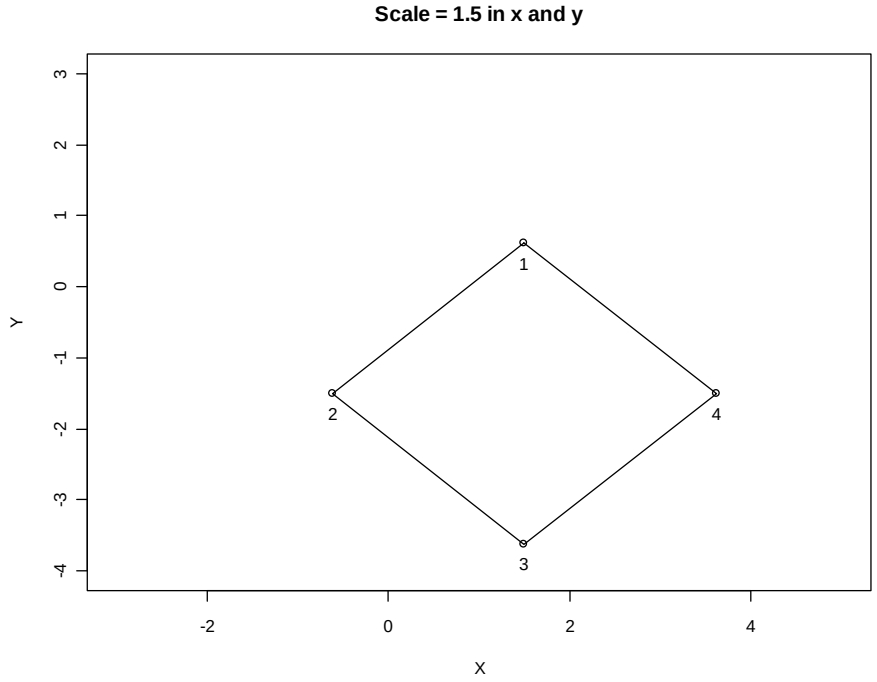


Table 2-7: Fit Results for the Translated, Rotated, and Equally Scaled Configuration.

	Procrustes	Euclidean	Affine
R^2	1	1	1
Translation in x	NA	1.5	1.5
Translation in y	NA	-1.5	-1.5
Scale in x	1.5	1.5	1.5
Scale in y	NA	NA	1.5
Rotation Angle	45	45	45
Shear Present	Unknown	Unknown	No

Figure 2-25: Translated, Rotated, and Unequally Scaled Configuration.

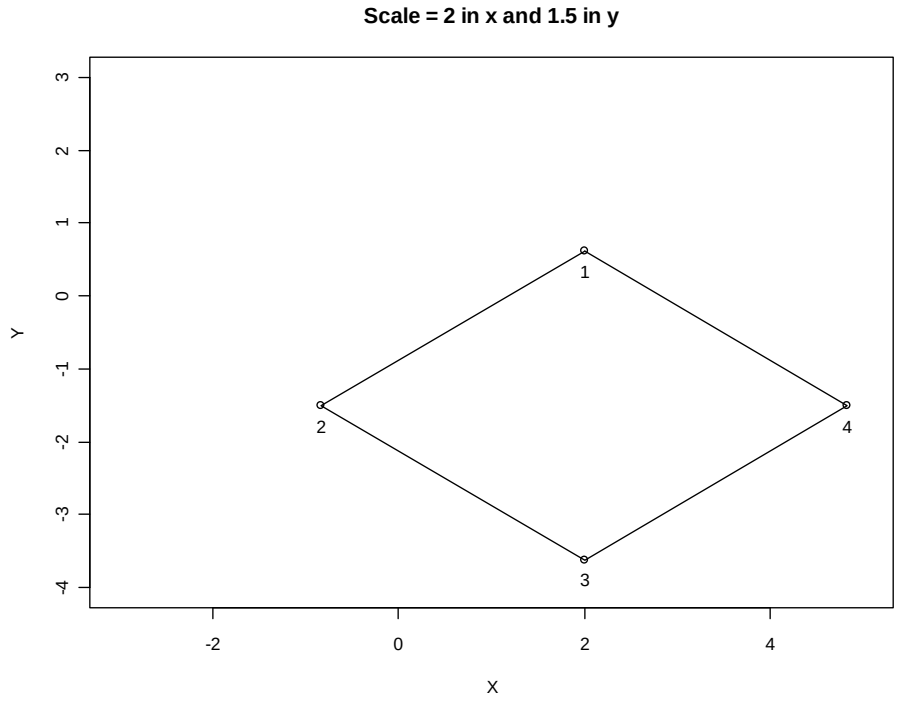


Table 2-8: Fit Results for the Translated, Rotated, and Unequally Scaled Configuration.

	Procrustes	Euclidean	Affine
R^2	0.98	0.98	1
Translation in x	NA	2	2
Translation in y	NA	-1.5	-1.5
Scale in x	1.73	1.75	2.0
Scale in y	NA	NA	1.5
Rotation Angle	36.87	45	48
Shear Present	Unknown	Unknown	Yes

Figure 2 -26: Translated, Rotated, Unequally Scaled, and Sheared Configuration.

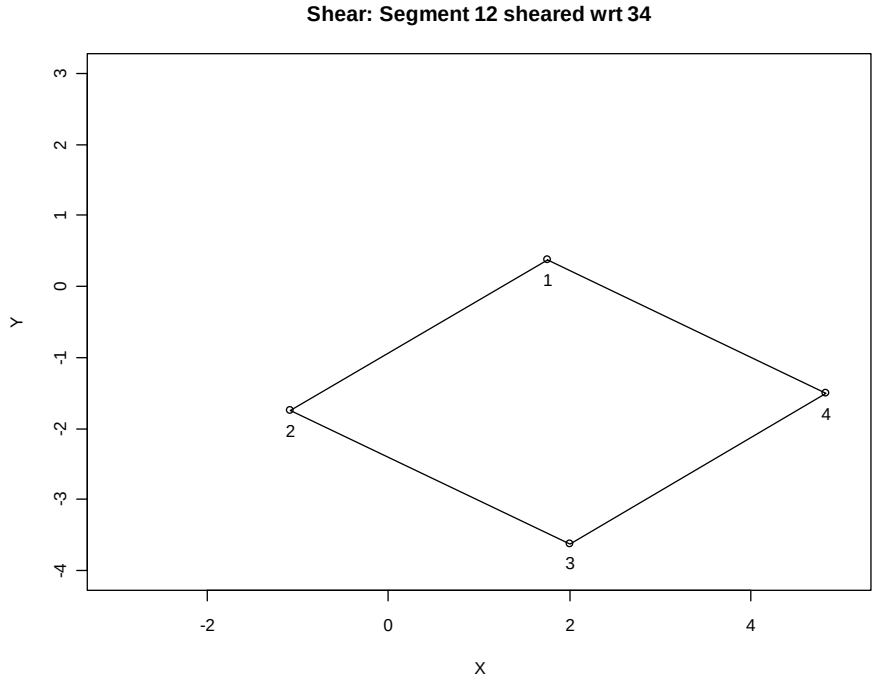


Table 2-5: Fit Results for Translated, Rotated, Unequally Scaled, and Sheared Configuration.

	Procrustes	Euclidean	Affine
R^2	0.96	0.98	1
Translation in x	NA	1.875	1.875
Translation in y	NA	-1.6	-1.625
Scale in x	1.72	1.75	2.09
Scale in y	NA	NA	1.41
Rotation Angle	36.87	47.89	48[6]
Shear Present	Unknown	Unknown	Yes

It is apparent from the rather simple example presented, that all three techniques perform well under known, controlled conditions; however, only the affine transformation gives the user the ability to detect possible shear, while still finding a "perfect" match.

2.4 Configuration Matching in Three-Dimensions

Obviously, there may be times where the sheer number of nodes in a given network will generate so many distances that the configuration will not be adequately represented in only one or two dimensions. Therefore, it would be helpful if the configuration matching techniques described in the previous section could be extended to more than two dimensions. Recalling the derivation of the Procrustes technique, at no point was the dimensionality of the input configurations specified. Meaning, the Procrustes technique is inherently extended to three dimensions and beyond since it will accept two matrices of any dimensionality.

[6] The rotation angle as calculated by equations 2.55a and b are not equal indicating shear is present. The angle reported is the average of the two calculated angles.

2.4.1 Extending the Euclidean Transformation to Three Dimensions

In their 2012 paper Schmid, et al extended the Euclidean, affine and projective transformations to three dimensions. For the Euclidean transformation, they looked at the problem as a series of three two dimensional Euclidean rotations; one about the x-axis, one about the y-axis, and one about the z-axis. The authors noted that while the rotation parameters will differ depending on the order in which the user rotates the configuration, the measure of fit (or similarity) stay the same. The three two dimensional rotation matrices are:

$$\boldsymbol{R}.\boldsymbol{x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}; \ \boldsymbol{R}.\boldsymbol{y} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}; \boldsymbol{R}.\boldsymbol{z} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Based on these rotation matrices, there are a total of six possible three dimensional Euclidean transformation matrices: x-y-z, x-z-y, y-x-z, y-z-x, z-x-y, z-y-x. Schmid, et al presented the x-y-z rotation matrix:

$$\boldsymbol{R} = [\boldsymbol{R}.\boldsymbol{x}][\boldsymbol{R}.\boldsymbol{y}][\boldsymbol{R}.\boldsymbol{z}] \quad (2.59)$$

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta\cos\phi & -\cos\theta\sin\phi & \sin\theta \\ \sin\gamma\sin\theta\cos\phi + \cos\gamma\sin\phi & -\sin\gamma\sin\theta\sin\phi + \cos\gamma\cos\phi & -\sin\gamma\cos\theta \\ -\cos\gamma\sin\theta\cos\phi + \sin\gamma\sin\phi & \cos\gamma\sin\theta\sin\phi + \sin\gamma\cos\phi & \cos\gamma\cos\theta \end{bmatrix}$$

The model thus becomes:

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} + s\boldsymbol{R} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \quad (2.60)$$

Where *s* is an overall scaling factor and

$$s\boldsymbol{R} = \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} \\ \beta_{21} & \beta_{22} & \beta_{23} \\ \beta_{31} & \beta_{32} & \beta_{33} \end{bmatrix} \quad (2.61)$$

Similar to the two dimensional derivation, vectorizing the new configurations and the parameter matrix and setting up the normal equations gives:

$$[\boldsymbol{X}'\boldsymbol{X}] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \beta_{11} \\ \beta_{12} \\ \beta_{13} \\ \beta_{21} \\ \beta_{22} \\ \beta_{23} \\ \beta_{31} \\ \beta_{32} \\ \beta_{33} \end{bmatrix} = \begin{bmatrix} \sum u_i \\ \sum v_i \\ \sum w_i \\ \sum u_i x_i \\ \sum u_i y_i \\ \sum u_i z_i \\ \sum v_i x_i \\ \sum v_i y_i \\ \sum v_i z_i \\ \sum w_i x_i \\ \sum w_i y_i \\ \sum w_i z_i \end{bmatrix} \quad (2.62)$$

Where ***X'X*** is:

$$\begin{bmatrix} N & 0 & 0 & \sum x_i & \sum y_i & \sum z_i & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N & 0 & 0 & 0 & 0 & \sum x_i & \sum y_i & \sum z_i & 0 & 0 & 0 \\ 0 & 0 & N & 0 & 0 & 0 & 0 & 0 & 0 & \sum x_i & \sum y_i & \sum z_i \\ \sum x_i & 0 & 0 & \sum x_i^2 & \sum x_i y_i & \sum x_i z_i & 0 & 0 & 0 & 0 & 0 & 0 \\ \sum y_i & 0 & 0 & \sum x_i y_i & \sum y_i^2 & \sum y_i z_i & 0 & 0 & 0 & 0 & 0 & 0 \\ \sum z_i & 0 & 0 & \sum x_i z_i & \sum y_i z_i & \sum z_i^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sum x_i & 0 & 0 & 0 & 0 & \sum x_i^2 & \sum x_i y_i & \sum x_i z_i & 0 & 0 & 0 \\ 0 & \sum y_i & 0 & 0 & 0 & 0 & \sum x_i y_i & \sum y_i^2 & \sum y_i z_i & 0 & 0 & 0 \\ 0 & \sum z_i & 0 & 0 & 0 & 0 & \sum x_i z_i & \sum y_i z_i & \sum z_i^2 & 0 & 0 & 0 \\ 0 & 0 & \sum x_i & 0 & 0 & 0 & 0 & 0 & 0 & \sum x_i^2 & \sum x_i y_i & \sum x_i z_i \\ 0 & 0 & \sum y_i & 0 & 0 & 0 & 0 & 0 & 0 & \sum x_i y_i & \sum y_i^2 & \sum y_i z_i \\ 0 & 0 & \sum z_i & 0 & 0 & 0 & 0 & 0 & 0 & \sum x_i z_i & \sum y_i z_i & \sum z_i^2 \end{bmatrix}$$

Equation 2.62 can now be solved to find the parameter estimates. The estimates of the overall scale; the angles of rotation; and the translations in each cardinal direction are all functions of the estimated scaled-rotation matrix parameter estimates and can be found as follows:

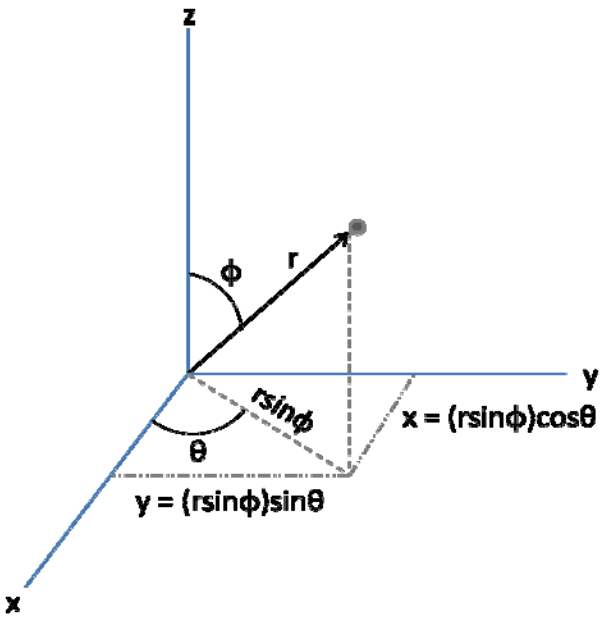
$$\hat{s} = \sqrt{\widehat{\beta_{11}}^2 + \widehat{\beta_{12}}^2 + \widehat{\beta_{13}}^2} \quad (2.63)$$

$$Rotation\ About\ x = \ \gamma = \tan^{-1}\left[\frac{\widehat{\beta_{23}}}{\widehat{\beta_{33}}}\right] \quad (2.64)$$

$$Rotation\ About\ y = \ \theta = \tan^{-1}\left[\frac{\widehat{\beta_{13}}}{\sqrt{\widehat{\beta_{11}}^2 + \widehat{\beta_{12}}^2}}\right] \quad (2.65)$$

One may be tempted to look for a singular rotation matrix by using spherical coordinates, instead of polar coordinates as adopted by Tobler and Schmid, et al. What follows is a short demonstration as to why the use of spherical coordinates is not viable. At first glance, the advantage to a spherical coordinates based transformation is that only ten parameters versus twelve parameters must be estimated. From geometry, a point in three dimensional space can be specified by its distance from the origin, ***r***; the angle between ***r*** and the z-axis, □; and the angle the image of ***r*** in the xy-plane makes with the x-axis, □ (see figure 2-19).

Figure 2 -27: Spherical Coordinates



From figure 2-19, the relationship between the spherical and Cartesian coordinate systems is as follows:

$$x = r\sin(\phi)\cos(\theta)$$

$$y = r\sin(\phi)\sin(\theta)$$

$$z = r\cos(\phi)$$

$$\phi = \tan^{-1}\frac{r\cos(\phi)}{r\sin(\phi)} = \frac{z}{\sqrt{x^2 + y^2}}$$

$$\theta = \tan^{-1}\frac{\sin(\theta)}{\cos(\theta)} = \tan^{-1}\frac{y}{x}$$

Now consider that a new configuration is found such that the point in figure 2-19 is closer to the xy-plane (i.e. z has decreased) and that the angle the image of ***r*** makes with the x-axis has increased; see figure 2-20. This is the same as increasing the angle □ by say δ and the angle □ by say λ, respectively. Note this is a rigid rotation, meaning the length of ***r*** has not changed.

Figure 2 - 28: New Configuration

The new Cartesian coordinates for the rotated point are:

$$u = r\sin(\phi + \delta)\cos(\theta + \lambda) = r\sin(\phi + \delta)[\cos\theta\cos\lambda - \sin\theta\sin\lambda]$$

$$v = r\sin(\phi + \delta)\sin(\theta + \lambda) = r\sin(\phi + \delta)[\sin\theta\cos\lambda + \cos\theta\sin\lambda]$$

$$w = r\cos(\phi + \delta)$$

Expanding the $\sin(\phi + \delta)$ and multiplying through by r yields:

$$u = x\cos\delta\cos\lambda - y\cos\delta\sin\lambda + z[\cos\theta\sin\delta\cos\lambda - \sin\theta\sin\delta\sin\lambda] \quad (2.68a)$$

$$v = y\cos\delta\cos\lambda + x\cos\delta\sin\lambda + z[\sin\theta\sin\delta\cos\lambda - \cos\theta\sin\delta\cos\lambda] \quad (2.68b)$$

$$w = z\ \cos\delta - \sqrt{x^2 + y^2}\sin\delta \quad (2.68c)$$

The problem with a rotation based on spherical coordinates is apparent in equation 2.68c. Under rigid rotation, the new coordinate *w* will be estimated to move in **only one** direction, either up or down.

To illustrate the problem, imagine three additional points are added to figure 2-20 (i.e. a three-dimensional pyramid) with one of the new points coincident with the x-axis. If the pyramid is to undergo a rigid rotation say about the x-axis, then the coordinates of the point that is coincident with the x-axis will not change; at least two of the new configurations points will have their "z value" decrease (increase) relative to their initial position; and the remaining point will have their "z-value" increase (decrease) relative to its initial position.

2.4.2 Three Dimensional Affine and Projective Transformations

The affine transformation and projective transformations were extended to three dimensions by Schmid, et. al. This was done by expanding Tobler's original 2D equations to include the third dimension. The affine transformation will be presented first.

$$u_i = \alpha_1 + \beta_1 x_i + \beta_2 y_i + \beta_3 z_i \quad (2.69a)$$

$$v_i = \alpha_2 + \beta_4 x_i + \beta_5 y_i + \beta_6 z_i \quad (2.69b)$$

$$w_i = \alpha_3 + \beta_7 x_i + \beta_8 y_i + \beta_9 z_i \quad (2.69c)$$

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} + \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \\ \beta_4 & \beta_5 & \beta_6 \\ \beta_7 & \beta_8 & \beta_9 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \quad (2.70)$$

Schmid, et. al. derives the 3D affine transformation by vectorizing the parameter matrix and solving using ordinary least squares, multiple-regression. As was demonstrated earlier, this can be treated as a multivariate, multiple-regression which allows for a more compact presentation and coding of the normal equations:

$$\begin{bmatrix} u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ \vdots & \vdots & \vdots \\ u_n & v_n & w_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & y_n & z_n \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_4 & \beta_7 \\ \beta_2 & \beta_5 & \beta_8 \\ \beta_3 & \beta_6 & \beta_9 \end{bmatrix} \quad (2.71)$$

$$\begin{bmatrix} \sum u_i & \sum v_i & \sum w_i \\ \sum x_i u_i & \sum x_i v_i & \sum x_i w_i \\ \sum y_i u_i & \sum y_i v_i & \sum y_i w_i \\ \sum z_i u_i & \sum z_i v_i & \sum z_i w_i \end{bmatrix} = \begin{bmatrix} n & \sum x_i & \sum y_i & \sum z_i \\ \sum x_i & \sum {x_i}^2 & \sum x_i y_i & \sum x_i z_i \\ \sum y_i & \sum x_i y_i & \sum {y_i}^2 & \sum y_i z_i \\ \sum z_i & \sum x_i z_i & \sum y_i z_i & \sum {z_i}^2 \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1 & \beta_4 & \beta_7 \\ \beta_2 & \beta_5 & \beta_8 \\ \beta_3 & \beta_6 & \beta_9 \end{bmatrix} \quad (2.72)$$

The estimated fit parameters can be estimated by solving equation 2.72 for $\boldsymbol{B}$. The same problem in interpreting the parameter meanings that arose in two dimensions arise in three dimensions, only now there are twice as many parameters to estimate and interpret.

The projective transformation introduces a vector of weights; where each element in the vector is the weight associated with each node. Recall that the affine transformation is a special case of the projective transformation where every element of the weighting vector is one. Again, Schmid et al expanded Tobler's original 2D equations to include the third dimension.

$$u_i = \frac{\alpha_1 + \beta_1 x_i + \beta_2 y_i + \beta_3 z_i}{\alpha_4 + \beta_{10} x_i + \beta_{11} y_i + \beta_{12} z_i} \quad (2.73a)$$

$$v_i = \frac{\alpha_2 + \beta_4 x_i + \beta_5 y_i + \beta_6 z_i}{\alpha_4 + \beta_{10} x_i + \beta_{11} y_i + \beta_{12} z_i} \quad (2.73b)$$

$$w_i = \frac{\alpha_3 + \beta_7 x_i + \beta_8 y_i + \beta_9 z_i}{\alpha_4 + \beta_{10} x_i + \beta_{11} y_i + \beta_{12} z_i} \quad (2.73c)$$

Again, just as with the 2D case, let $t = \alpha_4 + \beta_{10} x_i + \beta_{11} y_i + \beta_{12} z_i$ and rewrite equations 2.73a-c:

$$t_i u_i = \alpha_1 + \beta_1 x_i + \beta_2 y_i + \beta_3 z_i \quad (2.74a)$$

$$t_i v_i = \alpha_2 + \beta_4 x_i + \beta_5 y_i + \beta_6 z_i \quad (2.74b)$$

$$t_i w_i = \alpha_3 + \beta_7 x_i + \beta_8 y_i + \beta_9 z_i \quad (2.74c)$$

$$t_i = \alpha_4 + \beta_{10} x_i + \beta_{11} y_i + \beta_{12} z_i \quad (2.74d)$$

$$\begin{bmatrix} t_i u_i \\ t_i v_i \\ t_i w_i \\ t_i \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} + \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \\ \beta_4 & \beta_5 & \beta_6 \\ \beta_7 & \beta_8 & \beta_9 \\ \beta_{10} & \beta_{11} & \beta_{12} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \quad (2.75)$$

Using the same logic as before, this can be treated as a multivariate multiple-regression, which allows for a more compact presentation and coding of the normal equations:

$$\begin{bmatrix} t_1 u_1 & t_1 v_1 & t_1 w_1 & t_1 \\ t_2 u_2 & t_2 v_2 & t_2 w_2 & t_2 \\ \vdots & \vdots & \vdots & \vdots \\ t_n u_n & t_n v_n & t_i w_n & t_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & y_n & z_n \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \beta_1 & \beta_4 & \beta_7 & \beta_{10} \\ \beta_2 & \beta_5 & \beta_8 & \beta_{11} \\ \beta_3 & \beta_6 & \beta_9 & \beta_{12} \end{bmatrix} \quad (2.76)$$

$$\begin{bmatrix} \sum t_i u_i & \sum t_i v_i & \sum t_i w_i & \sum t_i \\ \sum x_i t_i u_i & \sum x_i t_i v_i & \sum x_i t_i w_i & \sum x_i t_i \\ \sum y_i\, t_i u_i & \sum y_i\, t_i v_i & \sum y_i t_i\, w_i & \sum y_i\, t_i \\ \sum z_i t_i u_i & \sum z_i t_i v_i & \sum z_i\, t_i w_i & \sum z_i\, t_i \end{bmatrix} = \begin{bmatrix} n & \sum x_i & \sum y_i & \sum z_i \\ \sum x_i & \sum {x_i}^2 & \sum x_i y_i & \sum x_i z_i \\ \sum y_i & \sum x_i y_i & \sum {y_i}^2 & \sum y_i\, z_i \\ \sum z_i & \sum x_i z_i & \sum y_i\, z_i & \sum {z_i}^2 \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \beta_1 & \beta_4 & \beta_7 & \beta_{10} \\ \beta_2 & \beta_5 & \beta_8 & \beta_{11} \\ \beta_3 & \beta_6 & \beta_9 & \beta_{12} \end{bmatrix}$$

The estimated fit parameters can be estimated by solving equation 2.76 for $\boldsymbol{B}$. Now there are sixteen parameters to be estimated, making the geometrical interpretation of the parameter estimates difficult at best.

Extending the affine and projective transformations beyond three dimensions is straight forward. Only now, there is no direct geometrical meaning to the parameter estimates. However, the only time the scale, rotation and reflection are important is if one is trying to match configurations based on true distances as measured in space. When dealing with true distances, the weighted adjacency matrix is saturated since there cannot be a situation where point A is y miles from point B; point B is z miles from point C; and point C is collocated with point A. On the other hand, a weighted adjacency matrix of communications can have the aforementioned property.

Most communications networks are sparsely populated. This means that a relatively large number of off diagonal elements are zero. In other words, a zero element represents two nodes that do not know each other, or who did not communicate during the time frame measured. Since the goal is to match how closely a communications network at time one matches the same network at time two, the only statistic of interest is the coefficient of determination. As networks get larger, the use of more than three

dimensions to get a configuration which most closely matches the communications levels may be needed.

2.5 Extending Configuration Matching to k –Dimensions

As previously stated, the Procrustes configuration matching technique is inherently extended to k-dimensions, and will not be discussed further. The Euclidean transformation is based on trigonometric functions that are defined in coordinate systems for which the reference axes are all mutually perpendicular. Therefore, extending the Euclidean transformation beyond three dimensions would require development of a new system of trigonometric functions for each additional dimension added; this is beyond the scope of this dissertation. The Euclidean transformation will not be considered in instances where the suggested dimensionality of a weighted adjacency matrix is more than three dimensions. This leaves extending the affine and projective transformations to k dimensions.

In two dimensions, the affine transformation required six parameters to be estimated and in three dimensions 12. If one adds a fourth dimension, the affine transformation would be:

$$w_{1i} = \alpha_1 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i}$$

$$w_{2i} = \alpha_2 + \beta_5 x_{1i} + \beta_6 x_{2i} + \beta_7 x_{3i} + \beta_8 x_{4i}$$

$$w_{3i} = \alpha_3 + \beta_9 x_{1i} + \beta_{10} x_{2i} + \beta_{11} x_{3i} + \beta_{12} x_{4i}$$

$$w_{4i} = \alpha_4 + \beta_{13} x_{1i} + \beta_{14} x_{2i} + \beta_{15} x_{3i} + \beta_{16} x_{4i}$$

for a total of 20 parameter estimates required to fit the model. In k-space, one needs to estimate a total of $k + k^2$ parameters. The general affine transformation has the form:

$$w_{1i} = \alpha_1 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \cdots + \beta_k x_{ki}$$

$$w_{2i} = \alpha_2 + \beta_{k+1} x_{1i} + \beta_{k+2} x_{2i} + \beta_{k+3} x_{3i} + \cdots + \beta_{2k} x_{ki}$$

$$\vdots$$

$$w_{ki} = \alpha_k + \beta_{k(k-1)+1} x_{1i} + \beta_{k(k-1)+2} x_{2i} + \beta_{k(k-1)+3} x_{3i} + \cdots + \beta_{k*k} x_{ki}$$

In matrix notation:

$$\begin{bmatrix} w_{1i} \\ w_{2i} \\ \vdots \\ w_{ki} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{bmatrix} + \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & \cdots & \beta_k \\ \beta_{k+1} & \beta_{k+2} & \beta_{k+3} & \cdots & \beta_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_{k(k-1)+1} & \beta_{k(k-1)+2} & \beta_{k(k-1)+3} & \cdots & \beta_{k(k)} \end{bmatrix} \begin{bmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{ki} \end{bmatrix} \quad (2.77)$$

The normal equations in multivariate multiple-regression matrix notation are:

1) The $\boldsymbol{W} = \boldsymbol{XB}$ analogue:

$$\begin{bmatrix} w_{11} & w_{21} & \cdots & w_{k1} \\ w_{12} & w_{22} & \cdots & w_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1n} & w_{2n} & \cdots & w_{kn} \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{k1} \\ 1 & x_{12} & x_{22} & \cdots & x_{k2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{kn} \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_k \\ \beta_1 & \beta_{k+1} & \cdots & \beta_{k(k-1)+1} \\ \beta_2 & \beta_{k+2} & \cdots & \beta_{k(k-1)+2} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_k & \beta_{2k} & \cdots & \beta_{k(k)} \end{bmatrix}$$

The $\boldsymbol{X'XB} = \boldsymbol{X'W}$ analogue:

$$\begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} & \cdots & \sum x_{ki} \\ \sum x_{1i} & \sum x_{1i}^2 & \sum x_{1i}x_{2i} & \cdots & \sum x_{1i}x_{ki} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum x_{2i}^2 & \cdots & \sum x_{2i}x_{ki} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_{ki} & \sum x_{1i}x_{ki} & \sum x_{2i}x_{ki} & \cdots & \sum x_{ki} \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_k \\ \beta_1 & \beta_{k+1} & \cdots & \beta_{k(k-1)+1} \\ \beta_2 & \beta_{k+2} & \cdots & \beta_{k(k-1)+2} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_k & \beta_{2k} & \cdots & \beta_{k(k)} \end{bmatrix}$$

$$= \begin{bmatrix} \sum w_{1i} & \sum w_{2i} & \cdots & \sum w_{ki} \\ \sum x_{1i}w_{1i} & \sum x_{1i}w_{2i} & \cdots & \sum x_{1i}w_{ki} \\ \sum x_{2i}w_{1i} & \sum x_{2i}w_{2i} & \cdots & \sum x_{2i}w_{ki} \\ \vdots & \vdots & \ddots & \vdots \\ \sum x_{ki}w_{1i} & \sum x_{ki}w_{2i} & \cdots & \sum x_{ki}w_{ki} \end{bmatrix}$$

This gives:

$$\widehat{\boldsymbol{B}} = \begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} & \cdots & \sum x_{ki} \\ \sum x_{1i} & \sum x_{1i}^2 & \sum x_{1i}x_{2i} & \cdots & \sum x_{1i}x_{ki} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum x_{2i}^2 & \cdots & \sum x_{2i}x_{ki} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_{ki} & \sum x_{1i}x_{ki} & \sum x_{2i}x_{ki} & \cdots & \sum x_{ki} \end{bmatrix}^{-1} \begin{bmatrix} \sum w_{1i} & \sum w_{2i} & \cdots & \sum w_{ki} \\ \sum x_{1i}w_{1i} & \sum x_{1i}w_{2i} & \cdots & \sum x_{1i}w_{ki} \\ \sum x_{2i}w_{1i} & \sum x_{2i}w_{2i} & \cdots & \sum x_{2i}w_{ki} \\ \vdots & \vdots & \ddots & \vdots \\ \sum x_{ki}w_{1i} & \sum x_{ki}w_{2i} & \cdots & \sum x_{ki}w_{ki} \end{bmatrix} \quad (2.78)$$

The affine transformation estimated coordinates, $\widehat{\boldsymbol{W}}$, are found in the usual manner:

$$\widehat{\boldsymbol{W}} = \boldsymbol{X}\widehat{\boldsymbol{B}} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}$$

The k-generalized fit statistic is then:

$$R^2_{k-affine} = 1 - \frac{\sum_{i=1}^{n}(w_{1i} - \widehat{w_{1i}})^2 + \cdots + \sum_{i=1}^{n}(w_{ki} - \widehat{w_{ki}})^2}{\sum_{i=1}^{n}(w_{1i} - \overline{w_{1.}})^2 + \cdots + \sum_{i=1}^{n}(w_{ki} - \overline{w_{1.}})^2}$$

In matrix notation this becomes:

$$R^2_{k-affine} = 1 - \frac{\sum\left\{Diag\left[(\boldsymbol{W} - \widehat{\boldsymbol{W}})'(\boldsymbol{W} - \widehat{\boldsymbol{W}})\right]\right\}}{\sum\{Diag[(\boldsymbol{W} - \overline{\boldsymbol{W}})'(\boldsymbol{W} - \overline{\boldsymbol{W}})]\}} \quad (2.79)$$

The generalization of the projective transformation is very similar to that of the affine, only now the vector of weights, T, must be incorporated. In the two dimensional projective case, there are nine parameters to be estimated and in three dimensions, there are 16. Adding a fourth dimension:

$$t_i w_{1i} = \alpha_1 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i}$$

$$t_i w_{2i} = \alpha_2 + \beta_5 x_{1i} + \beta_6 x_{2i} + \beta_7 x_{3i} + \beta_8 x_{4i}$$

$$t_i w_{3i} = \alpha_3 + \beta_9 x_{1i} + \beta_{10} x_{2i} + \beta_{11} x_{3i} + \beta_{12} x_{4i}$$

$$t_i w_{4i} = \alpha_4 + \beta_{13} x_{1i} + \beta_{14} x_{2i} + \beta_{15} x_{3i} + \beta_{16} x_{4i}$$

$$t_i = \alpha_5 + \beta_{17} x_{1i} + \beta_{18} x_{2i} + \beta_{19} x_{3i} + \beta_{20} x_{4i}$$

for a total of 25 parameters estimates required to fit the model. In k-space, one needs to estimate a total of $(k + 1)^2$ parameters. The linear models for each coordinate of the general projective transformation are:

$$t_i w_{1i} = \alpha_1 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \cdots + \beta_k x_{ki}$$

$$t_i w_{2i} = \alpha_2 + \beta_{k+1} x_{1i} + \beta_{k+2} x_{2i} + \beta_{k+3} x_{3i} + \cdots + \beta_{2k} x_{ki}$$

$$\vdots$$

$$t_i w_{ki} = \alpha_k + \beta_{k(k-1)+1} x_{1i} + \beta_{k(k-1)+2} x_{2i} + \beta_{k(k-1)+3} x_{3i} + \cdots + \beta_{k*k} x_{ki}$$

$$t_i = \alpha_{k+1} + \beta_{k*k+1}x_{1i} + \beta_{k*k+2}x_{2i} + \beta_{k*k+3}x_{3i} + \cdots + \beta_{k(k+1)}x_{ki}$$

In matrix notation:

$$\begin{bmatrix} t_i w_{1i} \\ t_i w_{2i} \\ \vdots \\ t_i w_{ki} \\ t_i \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \\ \alpha_{k+1} \end{bmatrix} + \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & \cdots & \beta_k \\ \beta_{k+1} & \beta_{k+2} & \beta_{k+3} & \cdots & \beta_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_{k(k-1)+1} & \beta_{k(k-1)+2} & \beta_{k(k-1)+3} & \cdots & \beta_{k(k)} \\ \beta_{k*k+1} & \beta_{k*k+2} & \beta_{k*k+3} & \cdots & \beta_{k(k+1)} \end{bmatrix} \begin{bmatrix} x_{1i} \\ x_{2i} \\ x_{3i} \\ \vdots \\ x_{ki} \end{bmatrix} \quad (2.80)$$

The normal equations in multivariate multiple-regression matrix notation are:

1) The $\boldsymbol{W}^* = \boldsymbol{XB}$ analogue[7]:

$$\boldsymbol{W}^* = \begin{bmatrix} t_1 w_{11} & t_1 w_{21} & \cdots & t_1 w_{k1} & t_1 \\ t_2 w_{12} & t_2 w_{22} & \cdots & t_2 w_{k2} & t_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ t_n w_{1n} & t_n w_{2n} & \cdots & t_n w_{kn} & t_n \end{bmatrix}$$

$$= \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{k1} \\ 1 & x_{12} & x_{22} & \cdots & x_{k2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & x_{2n} & \cdots & x_{kn} \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_k & \alpha_{k+1} \\ \beta_1 & \beta_{k+1} & \cdots & \beta_{k(k-1)+1} & \beta_{k*k+1} \\ \beta_2 & \beta_{k+2} & \cdots & \beta_{k(k-1)+2} & \beta_{k*k+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \beta_k & \beta_{2k} & \cdots & \beta_{k(k)} & \beta_{k(k+1)} \end{bmatrix}$$

The $\boldsymbol{X'XB} = \boldsymbol{X'W}^*$ analogue:

$$\begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} & \cdots & \sum x_{ki} \\ \sum x_{1i} & \sum x_{1i}^2 & \sum x_{1i}x_{2i} & \cdots & \sum x_{1i}x_{ki} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum x_{2i}^2 & \cdots & \sum x_{2i}x_{ki} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_{ki} & \sum x_{1i}x_{ki} & \sum x_{2i}x_{ki} & \cdots & \sum x_{ki} \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_k & \alpha_{k+1} \\ \beta_1 & \beta_{k+1} & \cdots & \beta_{k(k-1)+1} & \beta_{k*k+1} \\ \beta_2 & \beta_{k+2} & \cdots & \beta_{k(k-1)+2} & \beta_{k*k+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \beta_k & \beta_{2k} & \cdots & \beta_{k(k)} & \beta_{k(k+1)} \end{bmatrix}$$

[7] The star is used to distinguish that the weighted values for each coordinate in the new configuration is being fit by the model, versus the coordinate itself.

$$= \begin{bmatrix} \sum t_i w_{1i} & \sum t_i w_{2i} & \cdots & \sum t_i w_{ki} & \sum t_i \\ \sum x_{1i} t_i w_{1i} & \sum x_{1i} t_i w_{2i} & \cdots & \sum x_{1i} t_i w_{ki} & \sum x_{1i} t_i \\ \sum x_{2i} t_i w_{1i} & \sum x_{2i} t_i w_{2i} & \cdots & \sum x_{2i} t_i w_{ki} & \sum x_{2i} t_i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum x_{ki} t_i w_{1i} & \sum x_{ki} t_i w_{2i} & \cdots & \sum x_{ki} t_i w_{ki} & \sum x_{ki} t_i \end{bmatrix}$$

This gives:

$$\hat{\boldsymbol{B}} = \begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} & \cdots & \sum x_{ki} \\ \sum x_{1i} & \sum x_{1i}^2 & \sum x_{1i}x_{2i} & \cdots & \sum x_{1i}x_{ki} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum x_{2i}^2 & \cdots & \sum x_{2i}x_{ki} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_{ki} & \sum x_{1i}x_{ki} & \sum x_{2i}x_{ki} & \cdots & \sum x_{ki} \end{bmatrix}^{-1} \begin{bmatrix} \sum t_i w_{1i} & \sum t_i w_{2i} & \cdots & \sum t_i w_{ki} & \sum t_i \\ \sum x_{1i} t_i w_{1i} & \sum x_{1i} t_i w_{2i} & \cdots & \sum x_{1i} t_i w_{ki} & \sum x_{1i} t_i \\ \sum x_{2i} t_i w_{1i} & \sum x_{2i} t_i w_{2i} & \cdots & \sum x_{2i} t_i w_{ki} & \sum x_{2i} t_i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum x_{ki} t_i w_{1i} & \sum x_{ki} t_i w_{2i} & \cdots & \sum x_{ki} t_i w_{ki} & \sum x_{ki} t_i \end{bmatrix} \quad (2.81)$$

Let $\widehat{\boldsymbol{W}^*}$ be the weighted estimated coordinates, which are found in the usual last squares manner:

$$\widehat{\boldsymbol{W}^*} = \boldsymbol{X}\hat{\boldsymbol{B}} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}^*$$

The k-generalized fit statistic is then:

$$R^2_{k-projective} = 1 - \frac{\sum_{i=1}^{n}\left(w_{1i} - \widehat{w_{1i}^*}/t_i\right)^2 + \cdots + \sum_{i=1}^{n}\left(w_{ki} - \widehat{w_{ki}^*}/t_i\right)^2}{\sum_{i=1}^{n}(w_{1i} - \overline{w_{1.}})^2 + \cdots + \sum_{i=1}^{n}(w_{ki} - \overline{w_{1.}})^2}$$

Let $\widehat{\boldsymbol{W}} = \left\{\hat{w}_{ij} = \hat{w}_{ji}^*/\hat{t}_i\right\}$ then in matrix notation, $R^2_{k-projective}$ becomes:

$$R^2_{k-projective} = 1 - \frac{\sum\left\{Diag\left[\left(\boldsymbol{W} - \widehat{\boldsymbol{W}}\right)'\left(\boldsymbol{W} - \widehat{\boldsymbol{W}}\right)\right]\right\}}{\sum\{Diag[(\boldsymbol{W} - \overline{\boldsymbol{W}})'(\boldsymbol{W} - \overline{\boldsymbol{W}})]\}} \quad (2.82)$$

2.5.1 Behavior of affine k-dimensional coefficient of determination.

The behavior of the k-dimensional coefficient of determination will be explored in a similar fashion as the comparison between configuration matching techniques in section 2.3. To verify that the extension to k-dimensions is tractable and performs as expected, an initial matrix, X, of 1's and -1's was set up to represent a configuration in 4D space.

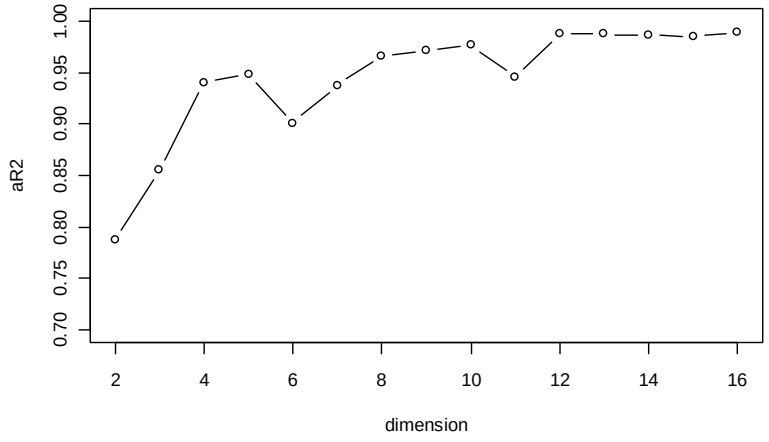
$$X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

The matrix X was then subsequently translated by 3 in the negative w direction, 2 in the positive x direction, 4 in the positive y direction, and 6 in the negative z directions. The matrix X was also sheared in the wy-plane by subtracting 3 from each of the lower 8 w-ordinates, and subtracting 2 from each of the lower 8 y-ordinates. The z-ordinate of the original matrix was also scaled by a factor of 4. Generalized affine regression was used to compare the original configuration to each of the manipulations described above, as well as each combination of the manipulations. In each case, the generalized aR^2 for the

comparison was 1. Rotation for 4D objects was not considered, since the extension of the trigonometric functions to more than 3D is beyond the scope of this dissertation.

Next, a symmetric 20x20 matrix was randomly generated, and the 2D to 16D configurations obtained using the cmdscale function in R environment. Each configuration size was matched to itself using generalized affine regression and in each case the aR^2 value was 1. A single edge was selected at random and its original value divided by 4. In this case, the edge between nodes 15 and 9 was replaced with 25% of its original value. It is expected that the value of aR2 will be less than 1 regardless of the configuration dimension used. The 2D, 3D,…, 16D MDS configurations were then compared to the original configuration. As with univariate multiple regression, it is expected that the value of the generalized aR^2 will increase as the number of dimensions in the configuration increases. Figure 2-21 contains a plot of aR^2 versus dimension for this scenario.

Figure 2-21: Plot of aR^2 vs dimension for a single edge reduced to 25% of its original value in a 20x20 network.



As expected, all of the aR^2 values obtained were less than 1. Inspection of figure 2-21 supports that, in general, aR^2 will increase as the number of dimensions used to describe the configuration increases. However, the value of aR^2 appears to start to level off between the 4D and 5D configurations; followed by a drop in aR^2 at for the 6D configurations. The value of aR^2 increases again from 6D to 10D configurations; with a leveling between 8D and 10D configurations followed by a drop at the 11D configurations. From 11D to 12D there is an increase in aR^2, and a leveling for all remaining dimensions around an aR^2 value of 0.98. The two drops in aR^2 at 6D and 11D are likely due to the discreteness of the edge weights.

A second edge (between the third and fifth nodes) was replaced by 3 times its original value; the same analysis as was done for the single edge perturbation was repeated here (figure 2-22). Here, the aR^2 value is expected to drop even further than for the single edge perturbation considered earlier.

Figure 2-22: Plot of aR^2 vs dimension for a second edge increased by 300% of its original value in a 20x20 network.

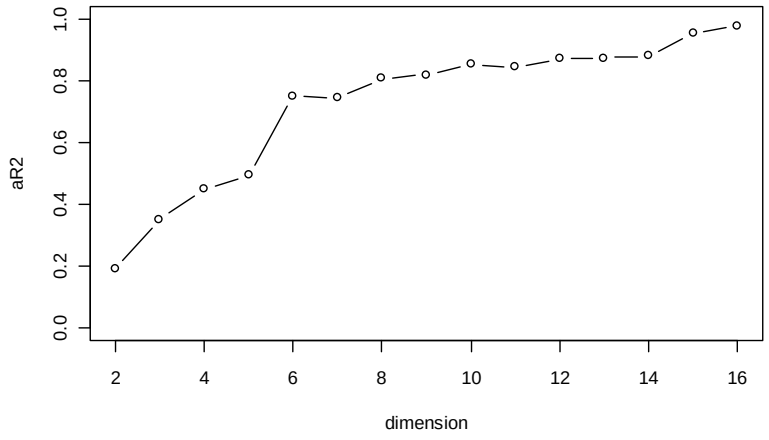


Figure 2-22, tends to reinforce that the generalized affine coefficient of determination behaves as expected. For all dimensions considered, the magnitude of the aR^2 for the single edge change is larger than the aR^2 for the two edge changes. However, when using 16D configurations the difference in aR^2 is only 0.01.

2.6 Conclusion

In this chapter, four configuration matching techniques were explored as to their suitability in matching configurations derived from distance matrices. All four techniques are designed to account for rigid body rotation, translation, and dilation. Excluding the Euclidean transformation, the three remaining techniques, procrustes, affine, and projective transformations were extended to k-dimensions. The procrustes technique was found to be limited in that it could only account for a single overall scaling factor and could not account for the presence of shear. On the other hand, the projective transformation is too flexible, in that it can account for fundamental position changes amongst the points of interest from one configuration to the next. The projective transformation also has the disadvantage of requiring a nonlinear estimation routine to be invoked to establish the individual weighting scheme for each observation. The ability of the affine transformation to account for rotation, reflection, translation, scale, and shear while remaining sensitive to relative position changes from one configuration to the next make it the best configuration matching technique for the purposes of this study.

2.7 References

Barlow, R. E. and Brunk, H. D. (1972), The Isotonic Regression Problem and its Dual, *Journal of the American Statistical Association*, Vol. 67, No. 337, 140-147.

Cox, T. F. and Cox, M. A. A. (2001), *Multidimensional Scaling, Second Edition*, CRC/Chapman and Hall, New York.

Householder, A. S. and Young, G. (1938), Matrix Approximation and Latent Roots, *The American Mathematical Monthly*, Vol. 45, No. 3, 165-171.

Johnson, R. A. and Wichern, D. W. (2007), *Applied Multivariate Statistical Analysis, Sixth Edition*, Pearson / Prentice Hall, New Jersey.

Kruskal, J. B. (1964b), Nonmetric Multidimensional Scaling: A Numerical, *Psychometrica*, 29, 115 – 129.

Mardia, K.V., Kent, J. T., and Bibby, J. M. (1979), *Multivariate Analysis*, Academic Press, London.

Moya, T. R. (2000), Calculating Isotonic Regression of the Distance Function in non-metric Multidimensional Scaling Model, *Methods of Psychological Research Online 2000*, Vol.5, No.3, 1 – 8.

R Development Core Team (2009), R: *A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Schmid, K. K., Marx, D. B., and Samal, S. (2012), Tridimensional Regression for Comparing and Mapping 3D Anatomical Structures, *Anatomy Research International,* Vol. 2012, 1-9.

Shepard, R. N. (1962), The analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function, Psychometrika, Vol 27, 125 – 139, 219 – 246.

Sibson, R (1978), Studies in the Robustness of Multidimensional Scaling: Procrustes Statistics, *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 40, No. 2, 234-238.

Tobler, W. (1994), Bidimensional Regression, *Geographical Analysis*, **26** (July), 186 – 212.

Torgerson, W. S. (1952), Multidimensional Scaling: I. Theory and Method, *Psychometrika*, Vol. 17, Issue 4, 401 – 409.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth Edition, Springer, New York.

Young, F.W. (1997), Multidimensional Scaling Lecture Notes, Professor Emeritus, Quantitative Psychology, University of North Carolina. http://forrest.psych.unc.edu.

Chapter 3

Detecting Changes in Closed Networks: A Simulation Study

In chapters 1 and 2, the proposed method for detecting a change in a closed network's communication volume involves obtaining the weighted adjacency matrix for sequential time periods; using metric multi-dimensional scaling to generate a configuration for each time period; sequentially matching configurations using bi- (or higher order) dimensional regression derived coefficients of determination; and finally comparing how well the sequential configurations match by comparing the coefficients of determination. To explore the validity of this proposal, a simulation study was conducted to determine the following:

1) Is the proposed method able to detect a change in a network based on a change in the number of communications between nodes?
2) If so, how small of a change can be detected?
3) Can the proposed method reliably detect "unexpected" differences in the network between time periods (i.e. what is the power of the test)?
4) Is the ability to detect a change influenced by the correlation of number of communications between nodes from time period to time period?
5) How sensitive is the proposed method to the number of dimensions used in obtaining a configuration?

In this study all simulated changes are done by replacing the number of communications on a randomly chosen edge with a multiple of itself. The multiple used

to simulate the change is called the perturbation factor. Prior to conducting the simulation study the following hypotheses were made regarding the five questions posed above. First, it is believed that for k out of m total non-zero edges (where $k < m$) the method will be able to detect a change in a closed network based solely on a change in the number of communications on the subset of k edges. For the purposes of grading the ability of the method to detect a change, any power below 0.6 will be considered poor; power between 0.6 and 0.8 will be considered moderate; and power between 0.8 and 1 will be considered good. With regard to questions 2 through 5, it is believed that when considering only one factor changing at a time power will increase as: the number of perturbed edges increases; the perturbation factor increases; the correlation increases; and the dimension of the configuration increases.

3.1 The Simulation

All simulations were done in the R statistical language environment. When necessary and/or for convenience, functions were written to accomplish specific tasks in the simulation. The functions and simulation code can be found in Appendix A. Functions written by the author are referenced in italics to differentiate them from functions that are provided in the R language. The functions that are contained in base R or an available R package will be italicized and underlined.

All simulations presented are based on ten node networks. Ten was chosen for two main reasons. First, consider a national government, although there are many entities (executive branch, legislative branch, taxation, military, law enforcement, education, etc)

that can be defined only a subset of these are likely to chosen to be in the analyst's network based on the question at hand. For example, in a deterrence scenario, the entities that define the closed network of interest might be executive, legislative, military, intelligence, and diplomatic, with some split into two or more entities depending on their function. Second, ten was chosen due to its manageable size and for its potential to provide non-trivial configurations in greater than two dimensions. A total of 52 time periods were generated for each of the 1000 replications of a single simulation. The i^{th} replication of a simulation starts by generating an adjacency matrix; once generated, the adjacency matrix remains unchanged for all 52 time periods. This requirement is in place, to ensure the proposed method is being tested on detecting changes based only on a shift in the number of communications between nodes.

There are many ways of simulating the adjacency matrix for a network (Snijders, 2006). In fact, the ergm package in R has a built in network simulation feature that allows the user to take into account node level and graph level indices (covariates) that can influence the presence or absence of an edge between two nodes (Hunter, et al, 2008). Since this work is focused on proof of concept, the simulation presented here is based on the simplest model described in Snijders, (2006) known as the Bernoulli model. A Bernoulli model is characterized by the probability that an edge exists between any two nodes is equal to the desired edge density of the network.

The presence or absence of an edge between any two nodes was randomly generated using a Bernoulli (p) distribution. Three values for p were considered: $p = 0.3$ ("low density"); $p = 0.6$ ("medium density"); $p = 0.9$ ("high density"). The minimum

edge density was set at 0.3 to try and avoid the potentially obtaining a linear or circular chain of edges. A linear chain is defined as node *A* connected to *B* and no other nodes; *B* is connected to *A* and *C* and no other nodes; *C* is connected to *B* and *D* and no others, etc. A circular chain is one in which node *A* can link to node *J* to complete the circle. With network size set at ten, there are 45 potential edges (symmetric network); a linear chain would have an edge density of 0.2. Therefore, an edge density of 0.3 was chosen as the minimum to avoid trivial 1 dimensional solutions. The function *edge.sim* in Appendix A was used to generate the adjacency matrices. The function has as inputs, n = number of nodes in the network, and p = probability an edge exists between any two nodes. Note: the output of *edge.sim* is an upper triangular matrix of 0s and 1s. This was done to reduce simulation run-time, and the full (symmetric) adjacency matrix can be found by simply adding the output of *edge.sim* is to its transpose.

With the current iteration's adjacency matrix established, the next step is to generate the 52 weighted adjacency matrices; each representing a snap-shot of the network at equally spaced time periods. To do this, McKenzie's (1988) first-order, autoregressive (AR(1)), Poisson process is used to generate the number of communications between distinct nodal pairs that share an edge. An AR(1) process was chosen to ensure a controllable level of correlation between the number of communications per nodal pair with time while still allowing fluctuations in the number of communications over time.

3.1.1 McKenzie's Derivation of an AR(1) Poisson Process

McKenzie begins his derivation by looking at the difference equation:

$$X_t = \rho * X_{t-1} + W_t = K_t + W_t \quad (3.1)$$

Where each X_{t-1} is independently and identically distributed (iid) Poisson (λ); each W_t is iid Poisson $\big((1-\rho)\lambda\big); 0 \leq \rho \leq 1$; each K_t, when given X_{t-1}, are independently distributed binomial (X_{t-1}, ρ); and X_{t-1} and W_t are independent. In the simulation, X_t is the number of communications at time t; X_{t-1} is the number of communications at time [t – 1]; $K_t|X_{t-1}$ is the binomial thinning of X_{t-1} at time t and W_t is a "fluctuation" term between times t and [t – 1].

To use equation 3.1, the unconditional distribution of K_t must be known. McKenzie uses an alternative form of the probability generating function to derive the unconditional distribution of K_t. First, define the alternative probability generating function (a.p.g.f.) for a random variable T as:

$E[(1-z)^T]; 0 \leq z \leq 1.$

For T distributed Poisson(λ), the a.p.g.f is:

$$E[(1-z)^T] = \sum_{T=0}^{\infty} (1-z)^T \frac{\lambda^T e^{-\lambda}}{T!}$$

$$= \sum_{T=0}^{\infty} [(1-z)\lambda]^T \frac{e^{-\lambda}}{T!}$$

$$= e^{-z\lambda} \sum_{T=0}^{\infty} [(1-z)\lambda]^T \frac{e^{-\lambda+z\lambda}}{T!}$$

$$= e^{-z\lambda}$$

For T distributed binomial (n, p), the a.p.g.f is:

$$E[(1-z)^T] = \sum_{T=0}^{n} \binom{n}{T} p^T (1-p)^{n-T} [(1-z)^T]$$

$$= \sum_{T=0}^{n} \binom{n}{T} [(1-z)p]^T (1-p)^{n-T}$$

$$= \big((1-z)p + 1 - p\big)^n$$

$$= (1-zp)^n$$

For convenience, the subscript has been dropped from both K_t and X_{t-1} in the derivation of the distribution of *K*. Given that $K|X$ is distributed binomial(X, ρ), the a.p.g.f. for K is:

$$E[(1-z)^K] = \sum_{X=0}^{\infty} E[(1-z)^K|X] \frac{\lambda^X e^{-\lambda}}{X!}$$

$$= \sum_{X=0}^{\infty} (1-z\rho)^X \frac{\lambda^X e^{-\lambda}}{X!}$$

$$= \sum_{X=0}^{\infty} \frac{\big((1-z\rho)\lambda\big)^X e^{-\lambda}}{X!}$$

$$= e^{-z\rho\lambda} \sum_{X=0}^{\infty} \frac{\left((1-z\rho)\lambda\right)^X e^{-\lambda+z\rho\lambda}}{X!}$$

$$= e^{-z\rho\lambda}$$

Which is the a.p.g.f. for a Poisson $(\rho\lambda)$ random variable; therefore K is distributed Poisson$(\rho\lambda)$. Now, since K_t and $W_t$are independent, their joint probability mass function is simply the product of their respective mass functions:

$$P(K,W) = \frac{(\rho\lambda)^K e^{-\rho\lambda}}{K!} \frac{[(1-\rho)\lambda]^W e^{-(1-\rho)\lambda}}{W!}$$

Now let $U = K + W$ and $V = K$, then:

$$P(U,V) = \frac{(\rho\lambda)^V e^{-\rho\lambda}}{V!} \frac{[(1-\rho)\lambda]^{(U-V)} e^{-(1-\rho)\lambda}}{(U-V)!}; 0 \le V \le U < \infty$$

Holding U constant and summing over V, the probability mass function of U is:

$$P(U) = \sum_{V=0}^{U} \frac{(\rho\lambda)^V e^{-\rho\lambda}}{V!} \frac{[(1-\rho)\lambda]^{(U-V)} e^{-(1-\rho)\lambda}}{(U-V)!}$$

$$= \frac{e^{-\lambda}}{U!} \sum_{V=0}^{U} \frac{U!\,(\rho\lambda)^V}{V!} \frac{[(1-\rho)\lambda]^{(U-V)}}{(U-V)!}$$

$$= \frac{e^{-\lambda}}{U!} \sum_{V=0}^{U} \binom{U}{V} (\rho\lambda)^V [(1-\rho)\lambda]^{(U-V)}$$

Applying the binomial theorem yields:

$$P(U) = \frac{e^{-\lambda}}{U!}[\rho\lambda + (1-\rho)\lambda]^U = \frac{\lambda^U e^{-\lambda}}{U!}.$$

So X_t as defined by equation 3.1 is distributed Poisson (λ). Notice, that X_t and X_{t-1} have the same unconditional distribution and thus, (3.1) is a λ mean stationary process. Additionally, the conditional expectation of X_t given X_{t-1} is $[X_t|X_{t-1}] = \rho X_{t-1} + (1-\rho)\lambda$.

Lastly, to show that (3.1) does generate an AR(1) process, the autocorrelation function for X_t will be derived by induction. First, examine the covariance between X_t and X_{t-1}:

$$cov(X_t X_{t-1}) = E[(X_t - \lambda)(X_{t-1} - \lambda)] = E[X_t X_{t-1} - \lambda X_t - \lambda X_{t-1} + \lambda^2]$$

$$= E[X_t X_{t-1}] - \lambda E[X_t] - \lambda E[X_{t-1}] + \lambda^2 = E[X_t X_{t-1}] - \lambda^2 \quad (3.2)$$

To evaluate the unconditional expectation, $E[X_t X_{t-1}]$, multiply both sides of (3.1) by X_{t-1} and take the expectation of the expression:

$$E[X_t X_{t-1}] = E[K_t X_{t-1} + W_t X_{t-1}] = E[E[K_t|X_{t-1}]X_{t-1} + W_t X_{t-1}]$$

$$= E[\rho X_{t-1}^2] + E[W_t X_{t-1}] \quad (3.3a)$$

Since W_t and X_{t-1} are independent and distributed Poisson $((1-\rho)\lambda)$ and Poisson(λ), respectively:

$$= \rho E[X_{t-1}^2] + E[W_t]E[X_{t-1}] = \rho(\lambda^2 + \lambda) + (1-\rho)\lambda^2 \quad (3.3b)$$

Substituting (3.3b) into (3.2) gives:

$$cov(X_t X_{t-1}) = \rho(\lambda^2 + \lambda) + (1-\rho)\lambda^2 - \lambda^2 = \rho\lambda \qquad (3.4)$$

Substituting (3.4) into the correlation equation:

$$cor(X_t X_{t-1}) = \frac{cov(X_t X_{t-1})}{\sqrt{var(X_t)var(X_{t-1})}} = \frac{cov(X_t X_{t-1})}{var(X_t)} = \frac{\rho\lambda}{\lambda} = \rho \quad (3.5)$$

Now consider the covariance between X_t and X_{t-2}.

$$cov(X_t X_{t-2}) = E[(X_t - \lambda)(X_{t-2} - \lambda)] = E[X_t X_{t-2} - \lambda X_t - \lambda X_{t-2} + \lambda^2]$$

$$= E[X_t X_{t-2}] - \lambda E[X_t] - \lambda E[X_{t-2}] + \lambda^2 = E[X_t X_{t-2}] - \lambda^2 \quad (3.6)$$

To evaluate $E[X_t X_{t-2}]$, multiply both sides of (3.1) by X_{t-2} and take the expectation of the expression:

$$E[X_t X_{t-2}] = E[K_t X_{t-2} + W_t X_{t-2}]$$

$$E[X_t X_{t-2}] = E[K_t X_{t-2}] + E[W_t X_{t-2}]$$

$$E[X_t X_{t-2}] = E[E[K_t | X_{t-1}] X_{t-2}] + E[W_t] E[X_{t-2}]$$

$$E[X_t X_{t-2}] = E[\rho X_{t-1} X_{t-2}] + (1-\rho)\lambda^2$$

$$E[X_t X_{t-2}] = \rho E[X_{t-1} X_{t-2}] + (1-\rho)\lambda^2 \quad (3.7)$$

Notice that if one takes the time as (t-1) versus t in equation 3.3a, $E[X_t X_{t-1}]$ becomes $E[X_{t-1} X_{t-2}]$ and thus:

$$E[X_{t-1} X_{t-2}] = \rho(\lambda^2 + \lambda) + (1-\rho)\lambda^2 = \rho\lambda + \lambda^2 \quad (3.8)$$

Substituting (3.8) into (3.7) gives:

$$E[X_t X_{t-2}] = \rho(\rho\lambda + \lambda^2) + (1-\rho)\lambda^2 = \rho^2\lambda + \lambda^2 \quad (3.9)$$

Substituting (3.9) into (3.6) gives:

$$cov(X_t X_{t-2}) = \rho^2\lambda + \lambda^2 - \lambda^2 = \rho^2\lambda \quad (3.10)$$

And thus:

$$cor(X_t X_{t-2}) = \frac{cov(X_t X_{t-2})}{\sqrt{var(X_t)var(X_{t-2})}} = \frac{cov(X_t X_{t-2})}{var(X_t)} = \frac{\rho^2\lambda}{\lambda} = \rho^2 \quad (3.11)$$

Notice that in (3.8) and (3.9) the power of ρ on the RHS is equal to the number of time periods separating the two random variables in the expectation on the LHS. Now assume that this relationship holds true for two random variables separated by (k-1) time periods, i.e.:

$$E[X_{t-1} X_{t-k}] = \rho^{k-1}\lambda + \lambda^2 \quad (3.12)$$

Then:

$$cov(X_t X_{t-k}) = E[X_t X_{t-k}] - \lambda^2 \quad (3.13)$$

$$E[X_t X_{t-k}] = E[K_t X_{t-k}] + E[W_t X_{t-k}] = E[E[K_t|X_{t-1}]X_{t-k}] + E[W_t]E[X_{t-k}]$$

$$= \rho E[X_{t-1} X_{t-k}] + (1-\rho)\lambda^2 \quad (3.14)$$

Substituting (3.12) into (3.14) gives:

$$E[X_t X_{t-k}] = \rho(\rho^{k-1}\lambda + \lambda^2) + (1-\rho)\lambda^2 = \rho^k\lambda + \lambda^2 \quad (3.15)$$

Substituting (3.15) into (3.13) gives:

$$cov(X_t X_{t-k}) = \rho^k \lambda \quad (3.16)$$

And thus the correlation between X_t and X_{t-k} is:

$$cor(X_t X_{t-2}) = \rho^k \quad (3.17)$$

Equation (3.17) is the autocorrelation function for an AR(1) process.

3.1.2 Simulating the Weighted Adjacency Matrices

With a Poisson AR(1) process established, the difference equation (3.1) can be used to generate a sequence of observations using the following algorithm:

1) Fix λ and ρ.

2) Generate a random X_0 from a Poisson (λ) distribution.

3) Generate a random $K_1|X_0$ from a binomial (X_0, ρ) distribution.

4) Generate a random W_1 from a Poisson ([1-ρ]λ) distribution.

5) Add the values found in steps 3 and 4 to find X_1

6) Replace X_0 in step 3 with the value found in step 5 and repeat.

This multi-step process starts with generating a matrix of initial Poisson means, λ_{ij}. Here λ_{ij} is the mean number of communications between the i^{th} and j^{th} nodes. The matrix of initial Poisson means is populated by sequentially stepping through the 45 upper triangular elements of the adjacency matrix, and randomly generating a

value (rounded to the nearest whole number) from a gamma (α, β) distribution. This task is accomplished by the *lambda* function at the start of a simulation and remains unchanged for all replications; *lambda* requires the following inputs: *n* = the number of nodes in the network; and *a* and *b*, are the α and 1/β gamma distribution parameter values, respectively.

The next step is to populate the time equal one weighted adjacency matrix. This step ensures that, at the start of a replication, the simulated time periods are based off of a different weighted adjacency matrix than the previous replication to avoid degeneracy. This task is accomplished by the function *mu.0* and utilizes the adjacency matrix for the current iteration, the number of nodes in the network, the correlation between time periods (ρ), and the matrix of initial Poisson means (λ_{ij}). For each non-zero ijth element (an edge) in the adjacency matrix, *mu.0* generates a new matrix where the ijth element is a value generated from a random Poisson ($\rho\lambda_{ij}$). The remaining weighted adjacency matrices for the replication are then populated using steps 3 through 6 of the Poisson AR(1) algorithm.

3.1.3 Simulating a Change

To simulate a change in the number of communications that is not consistent with the levels observed over time, the 26th and 52nd time periods in each replication were copied into their own separate files and perturbed by randomly selecting non-zero edges from the weighted adjacency matrix and multiplying each of the selected edge weights by the same constant, c. The values of c considered here are 1 (to establish type I error rate),

1.5, 2, 3.5, and 5. The functions *one.pert*, *two.perts*, *three.perts*, *four.perts*, and *five.perts* randomly perturb one, two, three, four, and five edges, respectively. There are two inputs to the *NUMBER.perts* functions; the weighted adjacency matrix to be perturbed and the size of the perturbation (the multiple used to increase or decrease the simulated number of communications associated with the selected edge).

3.2 Implementing and Evaluating the Procedure

With the simulated "normal condition" networks and the perturbed networks for two different time periods at hand, the evaluation of the proposed procedure can proceed. The first step is to find configurations for each simulated time period in each replication, and for the perturbed 26^{th} and 52^{nd} time periods. Normally, one would first do an eigen analysis of the squared, doubly centered adjacency matrix to determine the configuration dimension that best captures the true number of communications between nodes. Since one question of interest is to determine how sensitive the procedure is to the number of dimensions chosen for the configuration, this step is not included. For each time period, the network configuration in two dimensions was obtained using the function *wn.mds.* Inputs to the *wn.mds* function are the upper triangular weighted adjacency matrix, and the number of dimensions desired. The function first finds the symmetric weighted adjacency matrix by adding the upper triangular weighted adjacency matrix to its transpose. Next, a network configuration in k dimensions is found by passing the symmetric weighted adjacency matrix to the R function cmdscale (in the stats package of R). The function cmdscale implements Torgerson's metric multi-dimensional scaling algorithm that was described in chapter 1.

With the configurations for each time period established, the next step is to sequentially match the configurations using the affine transformation. The function *gen.affine* implements the generalized affine transformation developed in chapter 2. Inputs to the *gen.affine* function are the configurations for time periods t – 1 and t. Specifically, the affine coefficient of determination (aR^2) is found between time periods t = 0 and t = 1, then between period t = 1 and t = 2, then between periods t = 2 and t = 3, etc. The sequential aR^2's were found for the entire unperturbed simulation using a series of loops and these values were written to a file. The perturbed aR^2's were found by isolating the 25^{th} and 51^{st} unperturbed time periods and processing these with their respective perturbed time periods. To allow for the most flexibility in analysis, the perturbed aR^2's were written to a file. At this point, all the information necessary to detect a change in a network based on the number of communications between nodes has been simulated or calculated.

The foundation of the method for detecting a change in a network based on the number of communications between nodes is the ability to capture a series of "normal" communications periods before a change occurs. For the networks simulated here, the "normal" periods are times t = 2 to t = 25 and times t = 2 to t = 51 when the perturbation occurs at time t = 26 and t = 52, respectively. The first time period is excluded since it acts merely as the seed to the AR(1) process applied to each non-zero edge. The first (unperturbed) 24 (50) aR^2's are sorted and the 25^{th} (51^{st}) perturbed aR^2 is compared to the first (second) ordered unperturbed aR^2 value. The first (second) ordered aR2 values were chosen since they are the closest observed values to the stated level of 0.05. If the 25^{th}

(51^{st}) aR^2 is less than its respective ordered aR^2 then a change in the number of communications between nodes has been detected. The actual level of the test is 0.042 and 0.04 for perturbations occurring in the 26^{th} and 52^{nd} time periods, respectively. Based on these levels, it is expected that empirical type I errors will be between 0.029 to 0.054 and 0.028 to 0.052, for the 26^{th} and 52^{nd} times of perturbation, respectively. The type I error rate for two and three dimensional configurations at each combination of factors are provided in Tables 3-1 and 3-5. Inspection of both tables reveals that the type I error is within the expected range for each combination of factors.

Table 3-1: Type I error for tests based on 2D configurations.

Time of Perturbation	Number of Edges Perturbed	Edge Density	AR(1) Correlation Parameter					
			0.3		0.6		0.9	
			Type I Error	Std. Error	Type I Error	Std. Error	Type I Error	Std. Error
26 ($\alpha = 0.042$)	1	0.3	0.042	0.006	0.047	0.007	0.046	0.007
		0.6	0.044	0.006	0.039	0.006	0.037	0.006
		0.9	0.047	0.007	0.041	0.006	0.043	0.006
	3	0.3	0.044	0.007	0.027	0.005	0.039	0.006
		0.6	0.048	0.007	0.038	0.006	0.056	0.007
		0.9	0.049	0.007	0.051	0.007	0.053	0.007
	5	0.3	0.044	0.006	0.049	0.007	0.045	0.007
		0.6	0.037	0.006	0.038	0.006	0.037	0.006
		0.9	0.034	0.006	0.044	0.006	0.036	0.006
52 ($\alpha = 0.04$)	1	0.3	0.050	0.006	0.055	0.007	0.048	0.007
		0.6	0.047	0.007	0.038	0.006	0.045	0.007
		0.9	0.048	0.007	0.032	0.006	0.043	0.006
	3	0.3	0.053	0.007	0.038	0.006	0.036	0.006
		0.6	0.052	0.007	0.049	0.007	0.033	0.006
		0.9	0.039	0.006	0.036	0.006	0.041	0.006
	5	0.3	0.036	0.006	0.039	0.006	0.042	0.006
		0.6	0.041	0.006	0.047	0.007	0.040	0.006
		0.9	0.034	0.006	0.040	0.006	0.047	0.007

The empirical power was calculated by assigning a 1 to instances where the change was successfully detected (as defined above), and a 0 to each time a change was

not detected. The empirical power is then calculated by summing all of the 1's and 0's and dividing by the total number of replications:

$$Empirical\ Power = \frac{\sum I(success)}{No.\ Replications} \quad (3.18)$$

It should be noted here, that 1000 replications were used in each simulation. However, there were simulations based on edge densities of 0.3 that ended up with fewer than five non-zero edges. In these instances, a reduced number of replications were used in the five edge perturbation scenarios. Specifically, there were four instances where the number of replications was reduced to 999, and three instances where the number of replications was reduced to 998. The empirical powers in these cases were rounded to the nearest thousandth. This process was carried out for both two (2D) and three (3D) dimensional configurations for each simulation, edge perturbation, and perturbation factor combination. The 2D (section 3.2.1) and 3D (section 3.2.2) empirical power tables are presented in tables 3-3 through 3-4 and 3-6 through 3-8, respectively.

To aid in investigating the hypotheses made prior to conducting the study, plots of power versus the number of perturbed edges (for 2D and 3D configurations: figures 3-1 to 3-9 and figures 3-28 to 3-36, respectively); power versus perturbation factor (for 2D and 3D configurations: figures 3-10 to 3-18 and figures 3-37 to 3-45, respectively); and power versus AR(1) correlation parameter (for 2D and 3D configurations: figures 3-19 to 3-27 and figures 3-46 to 3-54, respectively) are provided. Wald 95% confidence intervals versus formal hypothesis tests are used to determine if there is evidence to support the hypotheses as stated and are included on each plot. In section 3.2.3, the fifth

hypothesis, power will increase with the number of dimensions, is explored using McNemar's test for paired proportions. The discussion of the affect of the number of observed historical time periods before a change can be found in chapter 4.

3.2.1Empirical Power for Two Dimensional Configurations

Table 3-2: Power for one edge perturbation at each combination of edge density, AR(1) correlation parameter, perturbation factor (PF) and time period of perturbation (TP).

TP	PF	Edge Density	AR(1) Correlation Parameter							
			0.3		0.6		0.9		1.0	
			Power	Std. Error	Power	Std. Error	Power	Std. Error	Power	Std. Error
26	1.5	0.3	0.132	0.011	0.190	0.012	0.397	0.015	1.000	-
		0.6	0.118	0.010	0.171	0.012	0.309	0.015	1.000	-
		0.9	0.095	0.009	0.096	0.009	0.208	0.013	1.000	-
	2	0.3	0.361	0.015	0.411	0.016	0.556	0.016	1.000	-
		0.6	0.258	0.014	0.281	0.014	0.499	0.016	1.000	-
		0.9	0.186	0.012	0.195	0.013	0.399	0.015	1.000	-
	3.5	0.3	0.605	0.015	0.665	0.015	0.720	0.014	1.000	-
		0.6	0.444	0.016	0.583	0.016	0.724	0.014	1.000	-
		0.9	0.619	0.015	0.630	0.015	0.663	0.015	1.000	-
	5	0.3	0.721	0.014	0.805	0.013	0.904	0.009	1.000	-
		0.6	0.720	0.014	0.785	0.013	0.847	0.011	1.000	-
		0.9	0.756	0.014	0.770	0.013	0.805	0.013	1.000	-
52	1.5	0.3	0.120	0.010	0.208	0.013	0.420	0.016	1.000	-
		0.6	0.121	0.010	0.182	0.012	0.326	0.015	1.000	-
		0.9	0.102	0.010	0.106	0.010	0.197	0.013	1.000	-
	2	0.3	0.372	0.015	0.420	0.016	0.525	0.016	1.000	-
		0.6	0.274	0.014	0.263	0.014	0.557	0.016	1.000	-
		0.9	0.171	0.012	0.210	0.013	0.381	0.015	1.000	-
	3.5	0.3	0.608	0.015	0.685	0.015	0.732	0.014	1.000	-
		0.6	0.462	0.016	0.584	0.016	0.688	0.015	1.000	-
		0.9	0.635	0.015	0.621	0.015	0.629	0.015	1.000	-
	5	0.3	0.746	0.014	0.791	0.013	0.895	0.010	1.000	-
		0.6	0.729	0.014	0.809	0.012	0.865	0.011	1.000	-
		0.9	0.718	0.014	0.778	0.013	0.818	0.012	1.000	-

Table 3-3: Power for three edge perturbations at each combination of edge density, AR(1) correlation parameter, perturbation factor (PF) and time period of perturbation (TP).

TP	PF	Edge Density	AR(1) Correlation Parameter							
			0.3		0.6		0.9		1.0	
			Power	Std. Error	Power	Std. Error	Power	Std. Error	Power	Std. Error
26	1.5	0.3	0.245	0.014	0.367	0.015	0.680	0.015	1.000	-
		0.6	0.226	0.013	0.266	0.014	0.633	0.015	1.000	-
		0.9	0.148	0.011	0.317	0.015	0.462	0.016	1.000	-
	2	0.3	0.540	0.016	0.654	0.015	0.834	0.012	1.000	-
		0.6	0.451	0.016	0.711	0.014	0.909	0.009	1.000	-
		0.9	0.513	0.016	0.531	0.016	0.739	0.014	1.000	-
	3.5	0.3	0.768	0.013	0.824	0.012	0.944	0.007	1.000	-
		0.6	0.828	0.012	0.882	0.010	0.971	0.005	1.000	-
		0.9	0.830	0.012	0.909	0.009	0.962	0.006	1.000	-
	5	0.3	0.814	0.012	0.879	0.010	0.935	0.008	1.000	-
		0.6	0.919	0.009	0.962	0.006	0.980	0.004	1.000	-
		0.9	0.972	0.005	0.969	0.005	0.944	0.007	1.000	-
52	1.5	0.3	0.234	0.013	0.344	0.015	0.675	0.015	1.000	-
		0.6	0.237	0.013	0.285	0.014	0.641	0.015	1.000	-
		0.9	0.136	0.011	0.321	0.015	0.454	0.016	1.000	-
	2	0.3	0.544	0.016	0.648	0.015	0.840	0.012	1.000	-
		0.6	0.471	0.016	0.739	0.014	0.896	0.00	1.000	-
		0.9	0.470	0.016	0.550	0.016	0.743	0.014	1.000	-
	3.5	0.3	0.799	0.013	0.824	0.012	0.925	0.008	1.000	-
		0.6	0.844	0.011	0.874	0.010	0.972	0.005	1.000	-
		0.9	0.831	0.012	0.909	0.009	0.963	0.006	1.000	-
	5	0.3	0.777	0.013	0.870	0.011	0.937	0.008	1.000	-
		0.6	0.932	0.008	0.971	0.005	0.976	0.005	1.000	-
		0.9	0.974	0.005	0.978	0.005	0.952	0.007	1.000	-

Table 3-4: Power for five edge perturbations at each combination of edge density, AR(1) correlation parameter, perturbation factor (PF) and time period of perturbation (TP).

TP	PF	Edge Density	AR(1) Correlation Parameter							
			0.3		0.6		0.9		1.0	
			Power	Std. Error	Power	Std. Error	Power	Std. Error	Power	Std. Error
26	1.5	0.3	0.260	0.014	0.403	0.016	0.687	0.015	1.000	-
		0.6	0.341	0.015	0.429	0.016	0.786	0.013	1.000	-
		0.9	0.176	0.012	0.497	0.016	0.582	0.016	1.000	-
	2	0.3	0.557	0.016	0.667	0.015	0.859	0.011	1.000	-
		0.6	0.630	0.015	0.701	0.014	0.926	0.008	1.000	-
		0.9	0.549	0.016	0.794	0.013	0.943	0.007	1.000	-
	3.5	0.3	0.779	0.013	0.809	0.012	0.922	0.008	1.000	-
		0.6	0.910	0.009	0.929	0.008	0.972	0.005	1.000	-
		0.9	0.882	0.010	0.971	0.005	0.988	0.003	1.000	-
	5	0.3	0.784	0.013	0.852	0.011	0.935	0.008	1.000	-
		0.6	0.895	0.010	0.948	0.007	0.973	0.005	1.000	-
		0.9	0.825	0.012	0.974	0.005	0.975	0.005	1.000	-
52	1.5	0.3	0.259	0.014	0.417	0.016	0.677	0.015	1.000	-
		0.6	0.340	0.015	0.433	0.016	0.800	0.013	1.000	-
		0.9	0.191	0.012	0.519	0.016	0.590	0.016	1.000	-
	2	0.3	0.568	0.016	0.666	0.015	0.852	0.011	1.000	-
		0.6	0.613	0.015	0.689	0.015	0.911	0.009	1.000	-
		0.9	0.536	0.016	0.802	0.013	0.946	0.007	1.000	-
	3.5	0.3	0.777	0.013	0.816	0.012	0.915	0.009	1.000	-
		0.6	0.922	0.008	0.929	0.008	0.973	0.005	1.000	-
		0.9	0.878	0.010	0.966	0.006	0.987	0.004	1.000	-
	5	0.3	0.762	0.013	0.843	0.012	0.924	0.008	1.000	-
		0.6	0.907	0.009	0.943	0.007	0.989	0.003	1.000	-
		0.9	0.831	0.012	0.975	0.005	0.990	0.003	1.000	-

There are four immediate observations that can be made from inspection of the 2D empirical power tables (Tables 3-2 to 3-4). First, for single edge and three edge perturbations and perturbation factors of 1.5 and 2, power for detecting a change is poor. The exceptions to this occurred when there were three edge perturbations and the AR(1) correlation was high. Second, in general, power stayed the same or increased as the number of "normal" time periods observed (before a change) increases. Third, as predicted it appears that power and the AR(1) correlation parameter are positively correlated. Fourth, for a given AR(1) correlation parameter, power increases as the perturbation factor increases.

3.2.1.1 Power versus the Number of Edges Perturbed

Figure 3-1: Plots of empirical power vs. number of edges perturbed for edge density = 0.3 and AR(1) correlation parameter = 0.3; 2D configuration.

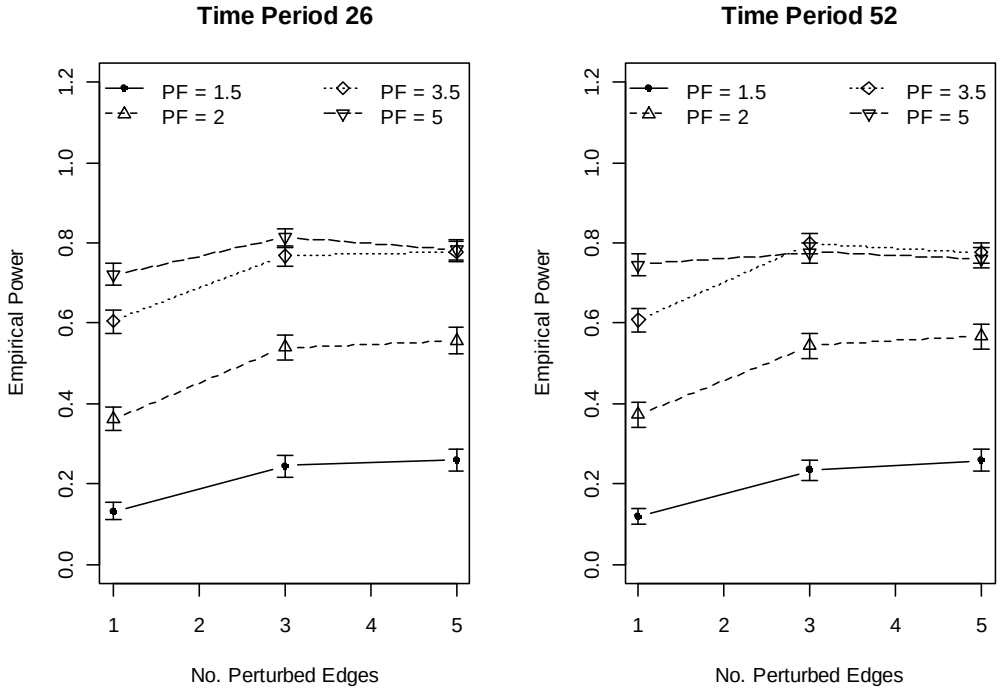


Figure 3-2: Plots of empirical power vs. number of edges perturbed for edge density = 0.3 and AR(1) correlation parameter = 0.6; 2D configuration.

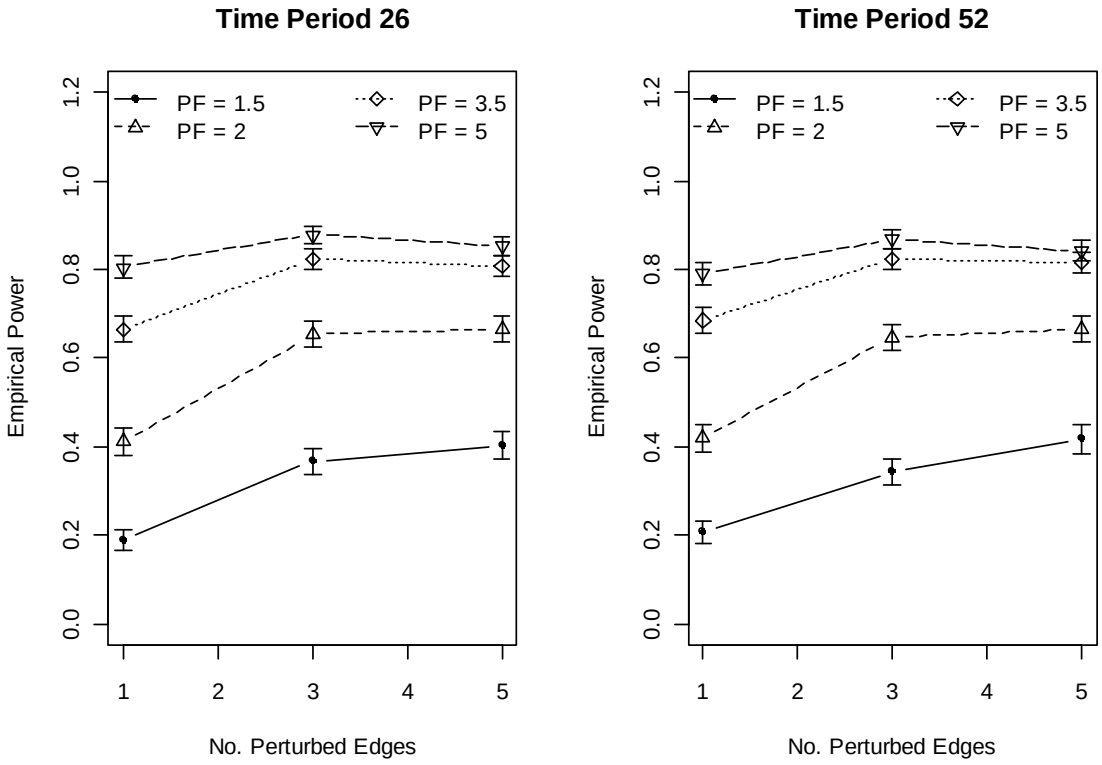


Figure 3-3: Plots of empirical power vs. number of edges perturbed for edge density = 0.3 and AR(1) correlation parameter = 0.9; 2D configuration.

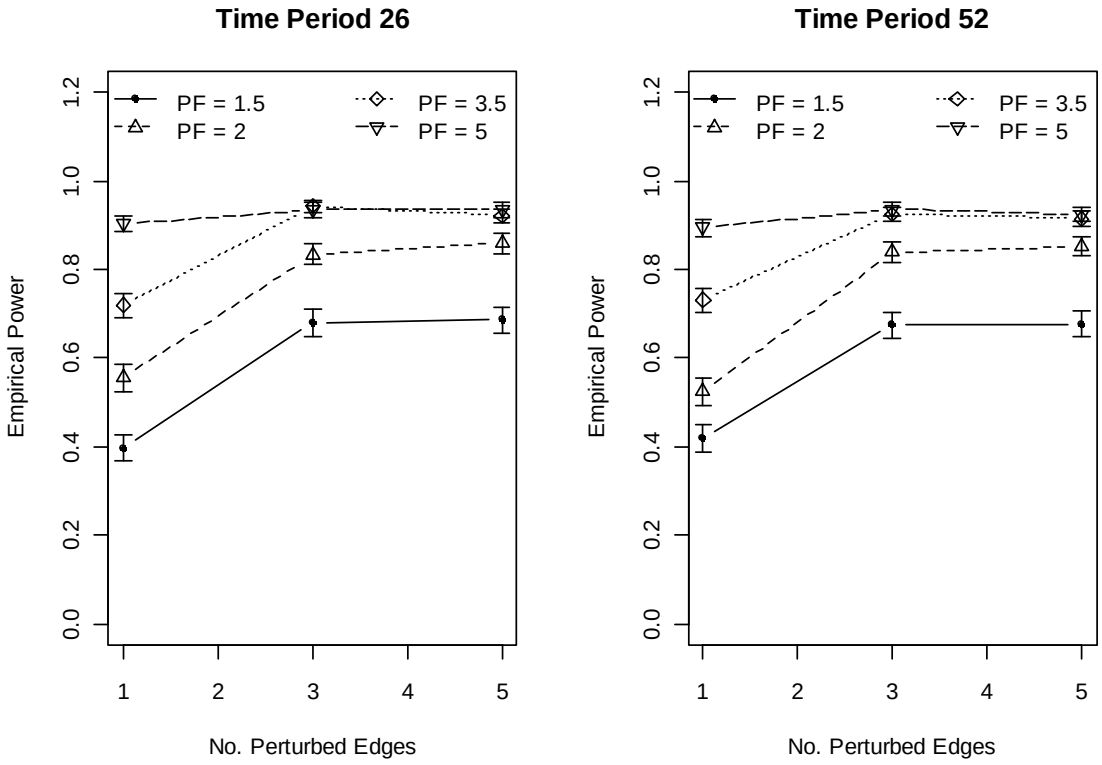


Figure 3-4: Plots of empirical power vs. number of edges perturbed for edge density = 0.6 and AR(1) correlation parameter = 0.3; 2D configuration.

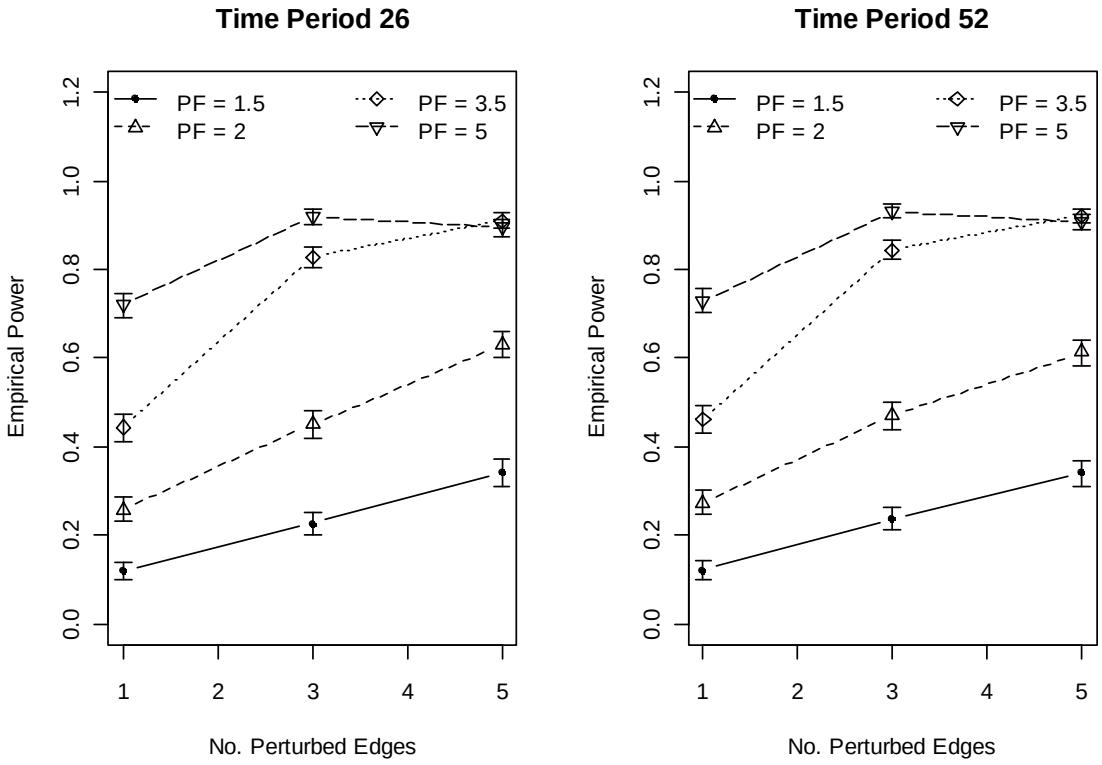


Figure 3-5: Plots of empirical power vs. number of edges perturbed for edge density = 0.6 and AR(1) correlation parameter = 0.6; 2D configuration.

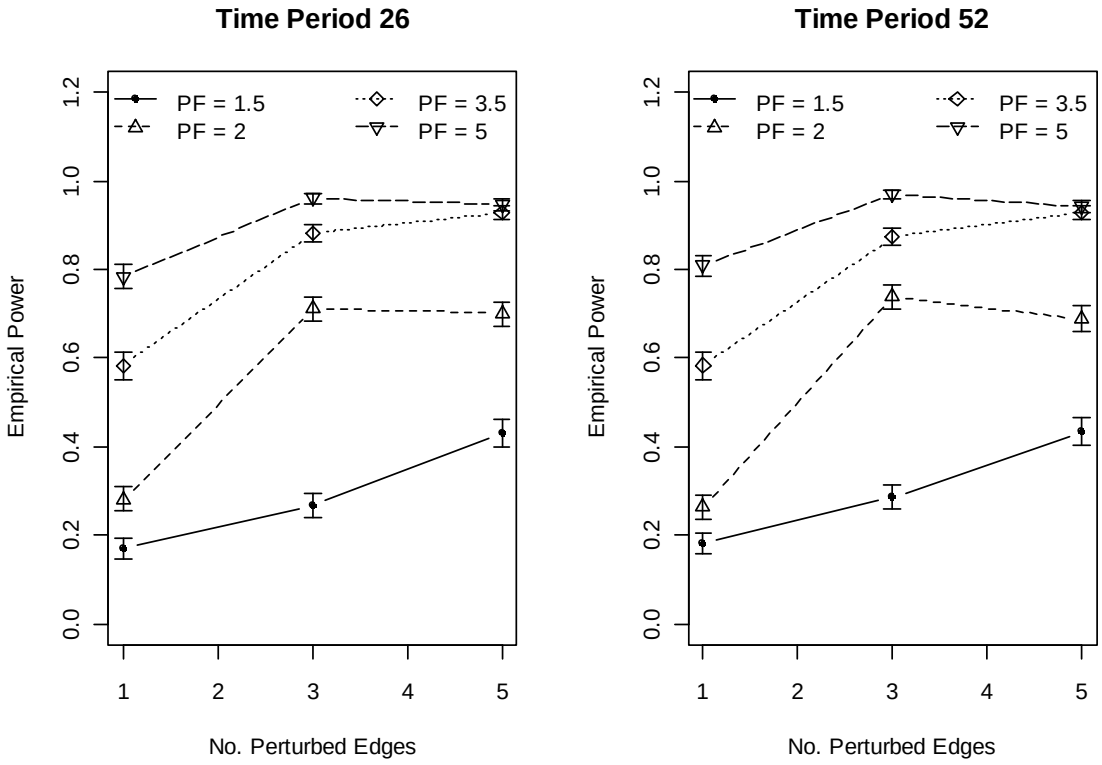


Figure 3-6: Plots of empirical power vs. number of edges perturbed for edge density = 0.6 and AR(1) correlation parameter = 0.9; 2D configuration.

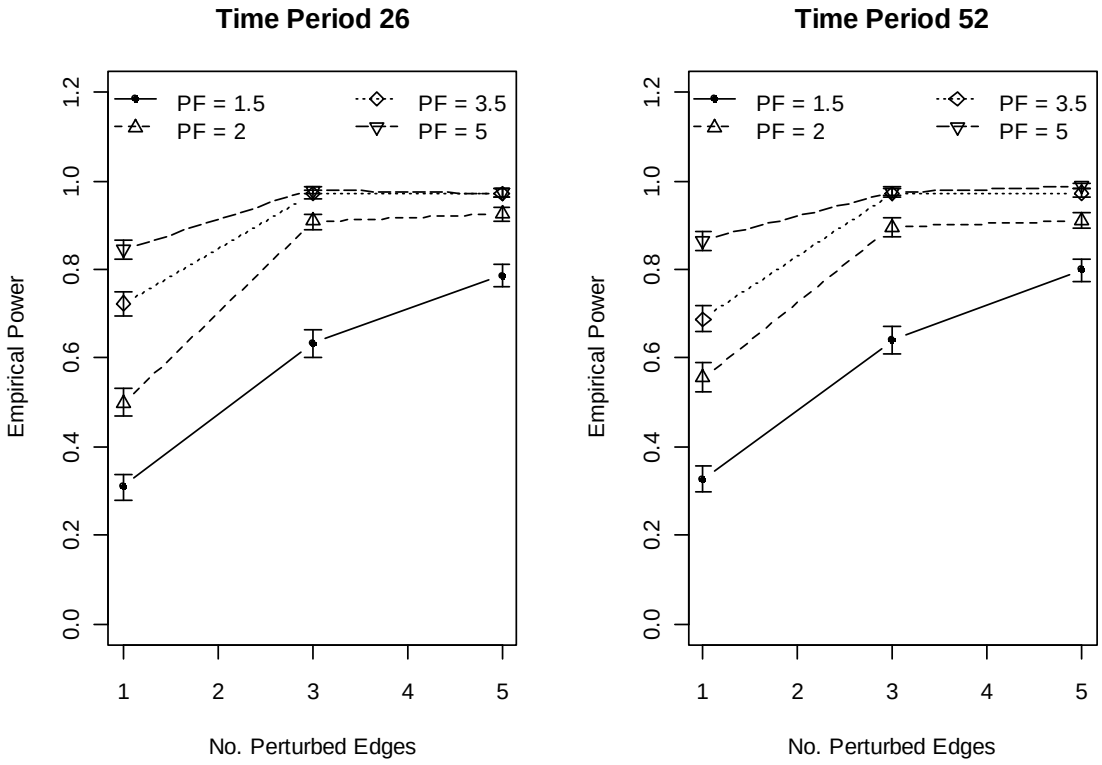


Figure 3-7: Plots of empirical power vs. number of edges perturbed for edge density = 0.9 and AR(1) correlation parameter = 0.3; 2D configuration.

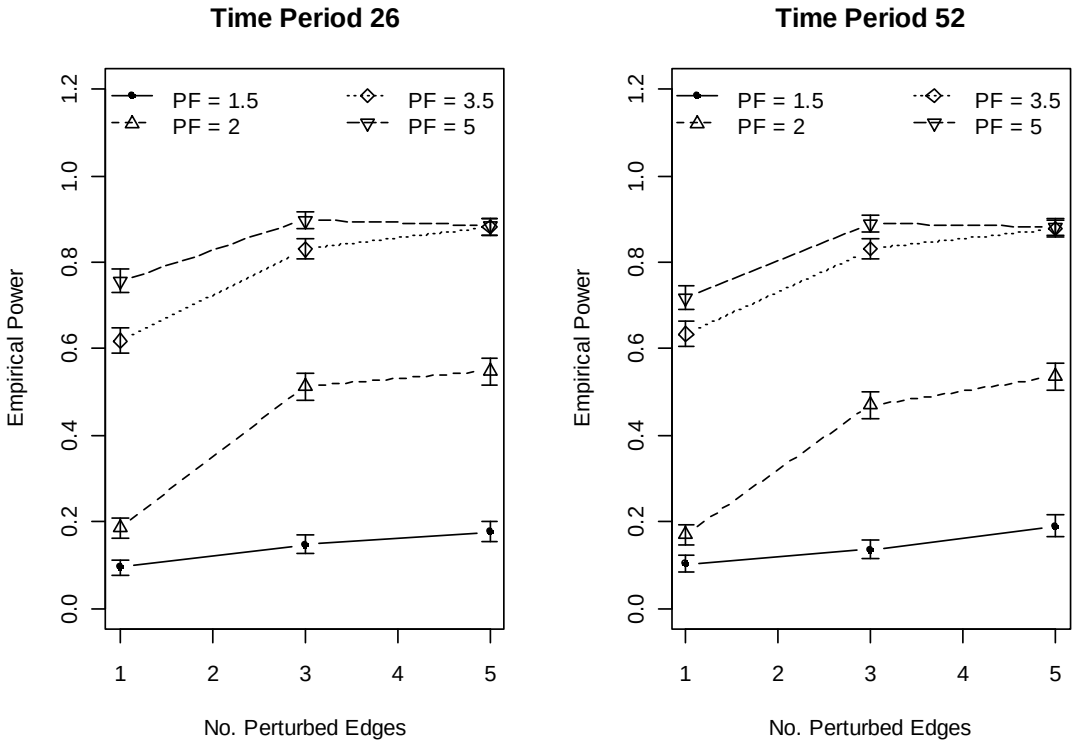


Figure 3-8: Plots of empirical power vs. number of edges perturbed for edge density = 0.9 and AR(1) correlation parameter = 0.6; 2D configuration.

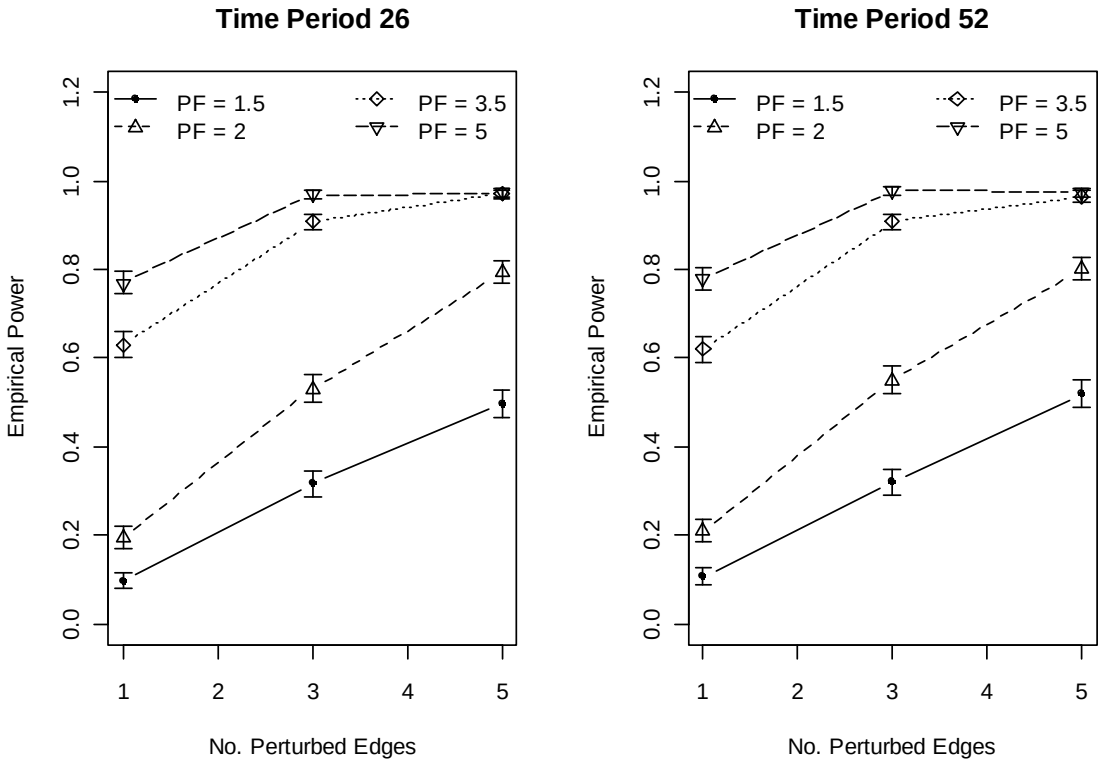
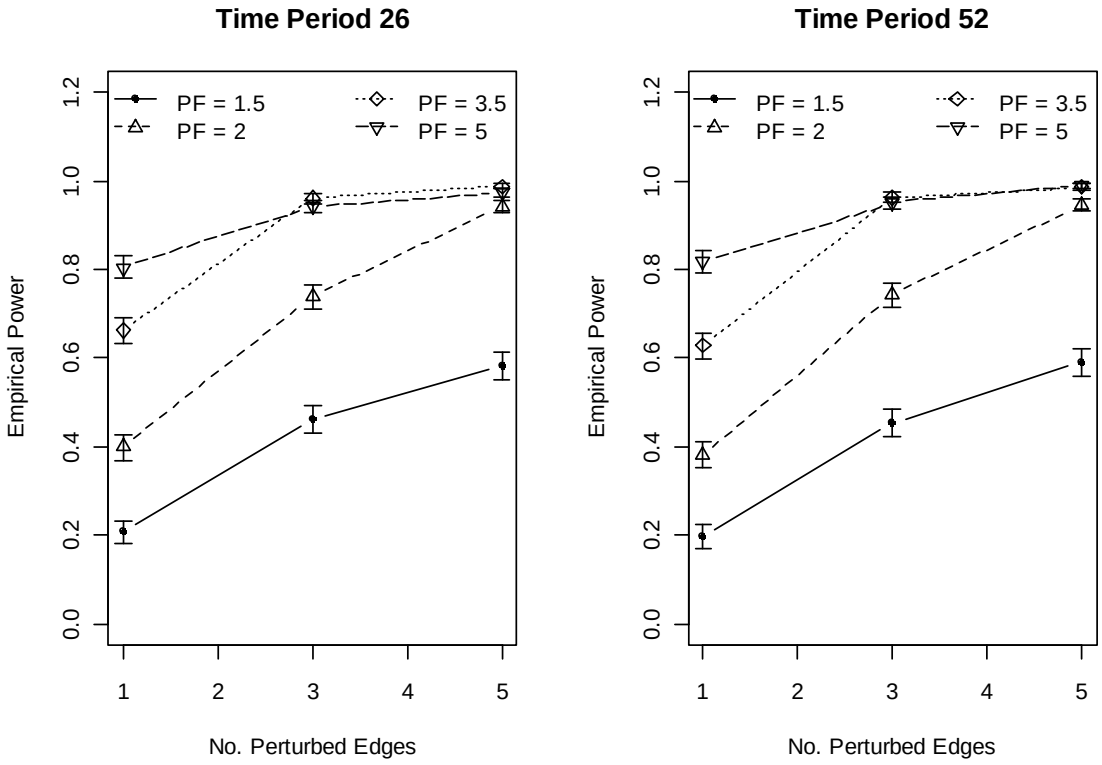


Figure 3-9: Plots of empirical power vs. number of edges perturbed for edge density = 0.9 and AR(1) correlation parameter = 0.9; 2D configuration.



Inspection of figures 3-1 to 3-9 reveals that the shape of the power versus number of perturbed edges are the very similar, regardless of edge density. Specifically, for a perturbation factor of 1.5, power remains right at or below 0.060 regardless of the edge density, AR(1) correlation parameter combination. For a perturbation factor of 2, power is at or above 0.060 for all edge densities when the AR(1) correlation parameter is 0.6 or 0.9 and the number of edges perturbed is 3 or more. For combinations of perturbation factor 3.5 and 5 and number of perturbed edges 3 and 5, the confidence intervals for power overlap for all combinations of the remaining factors. In all cases, power

increased or stayed within the adjacent margin of error as the number of perturbed edges increased.

3.2.1.2 Power versus Perturbation Factor

Figure 3-10: Plots of empirical power vs. perturbation factor for edge density = 0.3 and a single edge perturbation; 2D configuration.

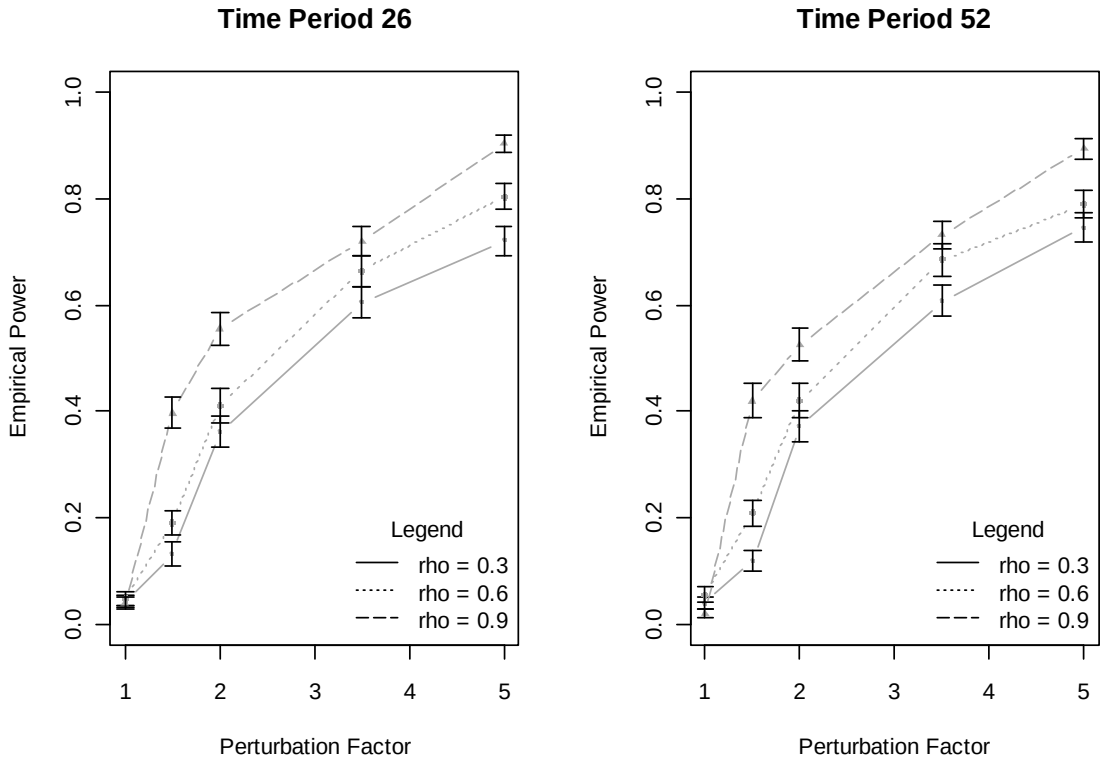


Figure 3-11: Plots of empirical power vs. perturbation factor for edge density = 0.3 and three edge perturbations; 2D configuration.

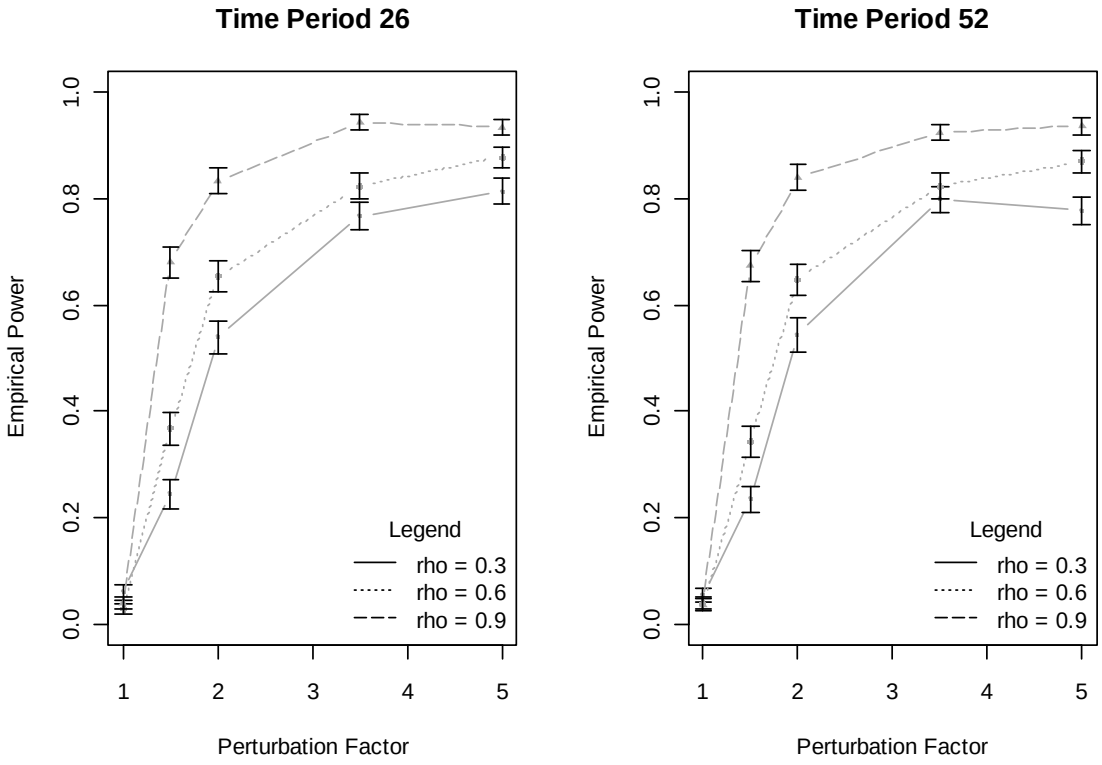


Figure 3-12: Plots of empirical power vs. perturbation factor for edge density = 0.3 and five edge perturbations; 2D configuration.

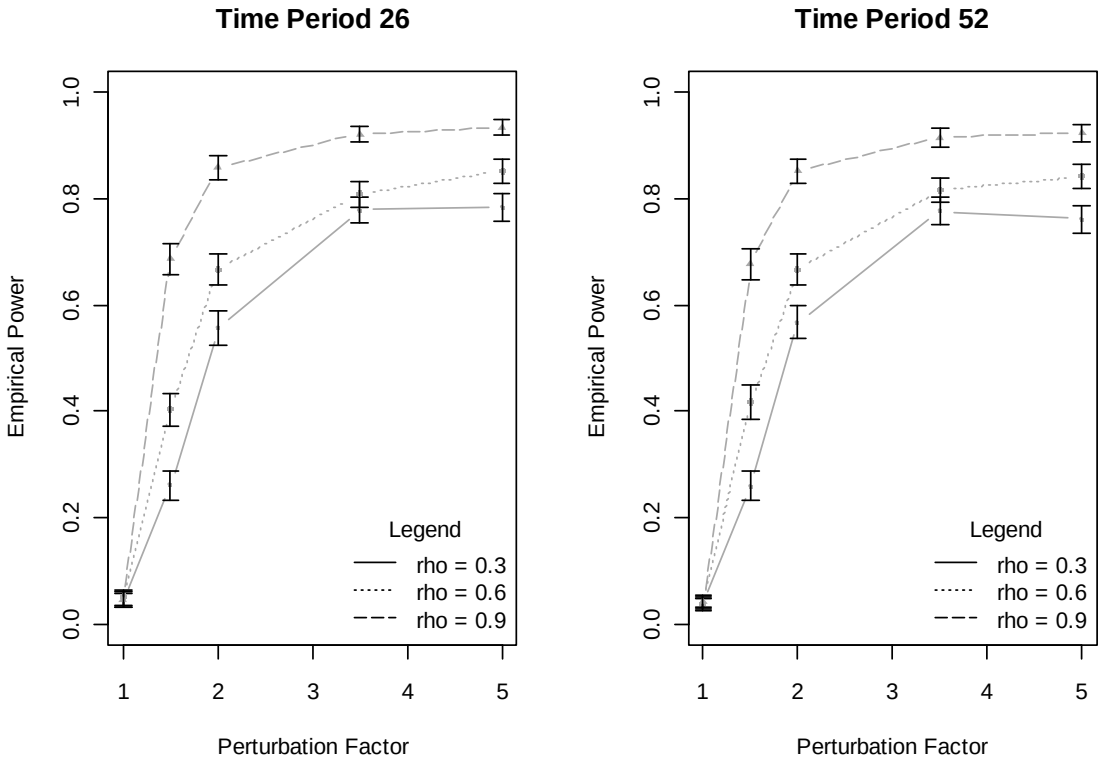


Figure 3-13: Plots of empirical power vs. perturbation factor for edge density = 0.6 and a single edge perturbation; 2D configuration.

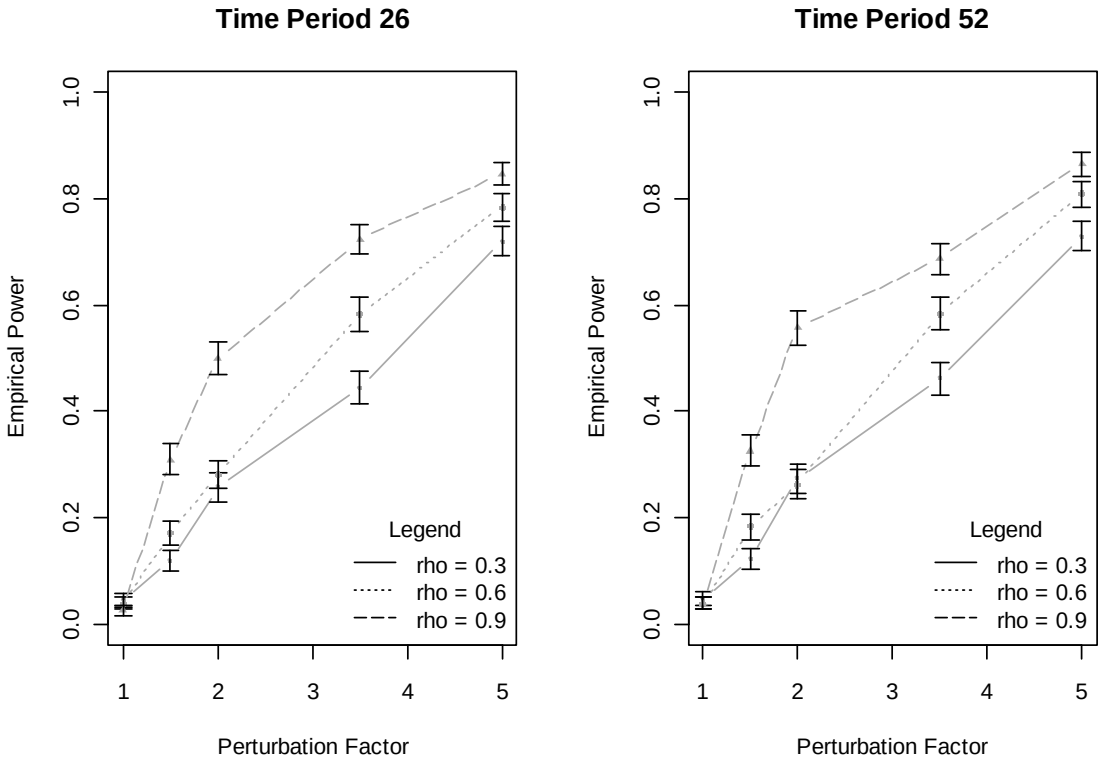


Figure 3-14: Plots of empirical power vs. perturbation factor for edge density = 0.6 and three edge perturbations; 2D configuration.

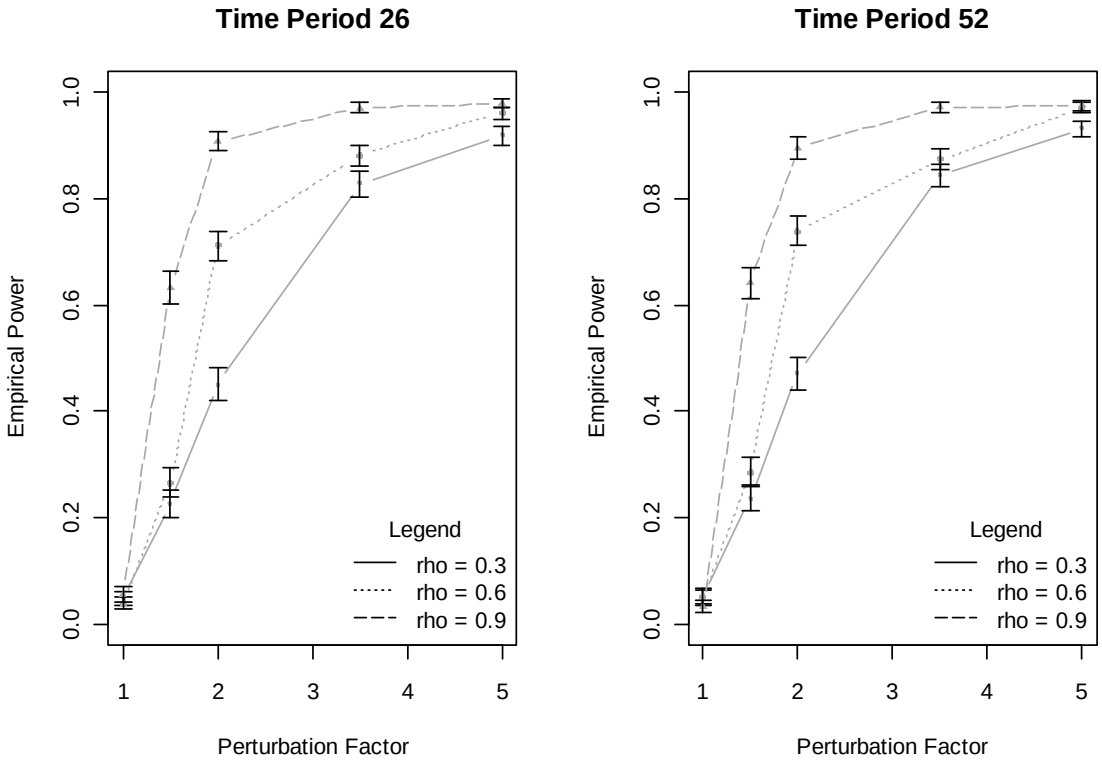


Figure 3-15: Plots of empirical power vs. perturbation factor for edge density = 0.6 and five edge perturbations; 2D configuration.

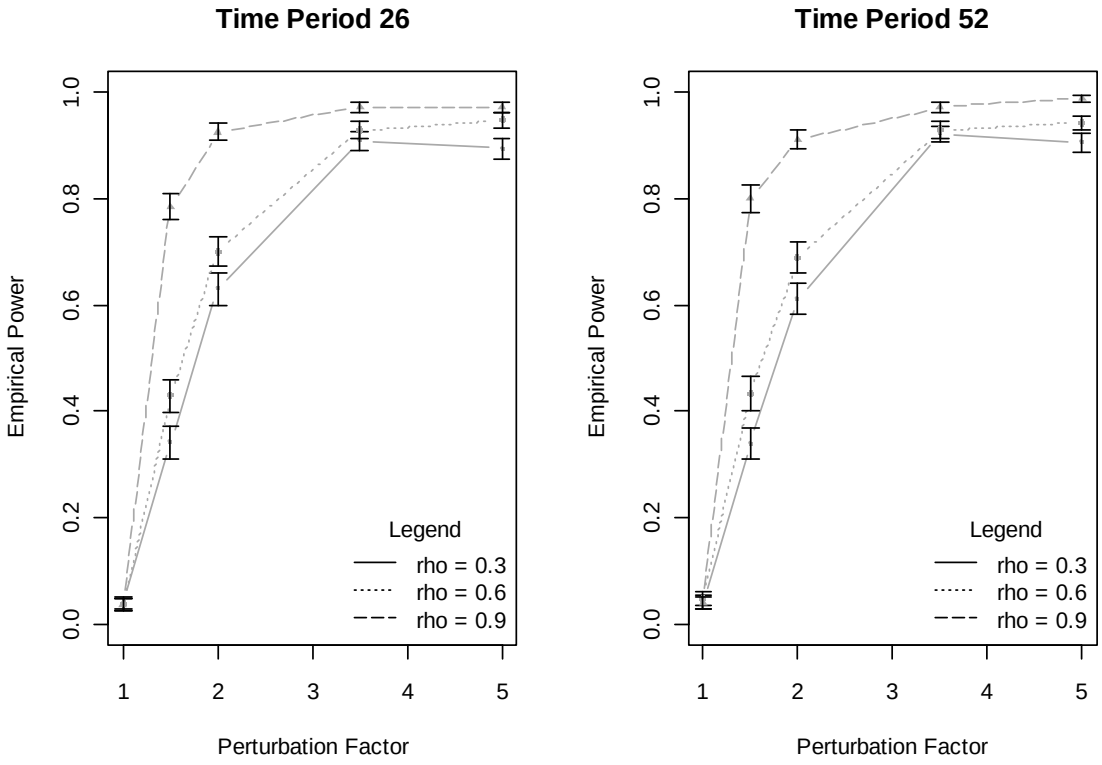


Figure 3-16: Plots of empirical power vs. perturbation factor for edge density = 0.9 and a single edge perturbation; 2D configuration.

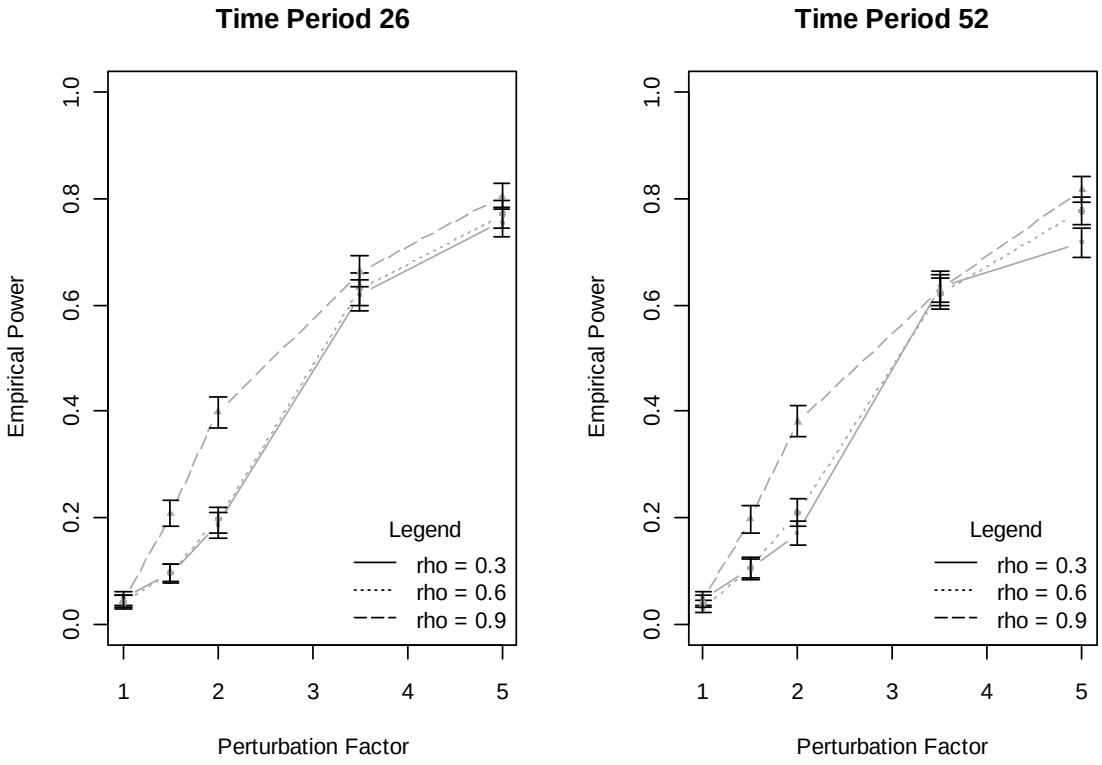


Figure 3-17: Plots of empirical power vs. perturbation factor for edge density = 0.9 and three edge perturbations; 2D configuration.

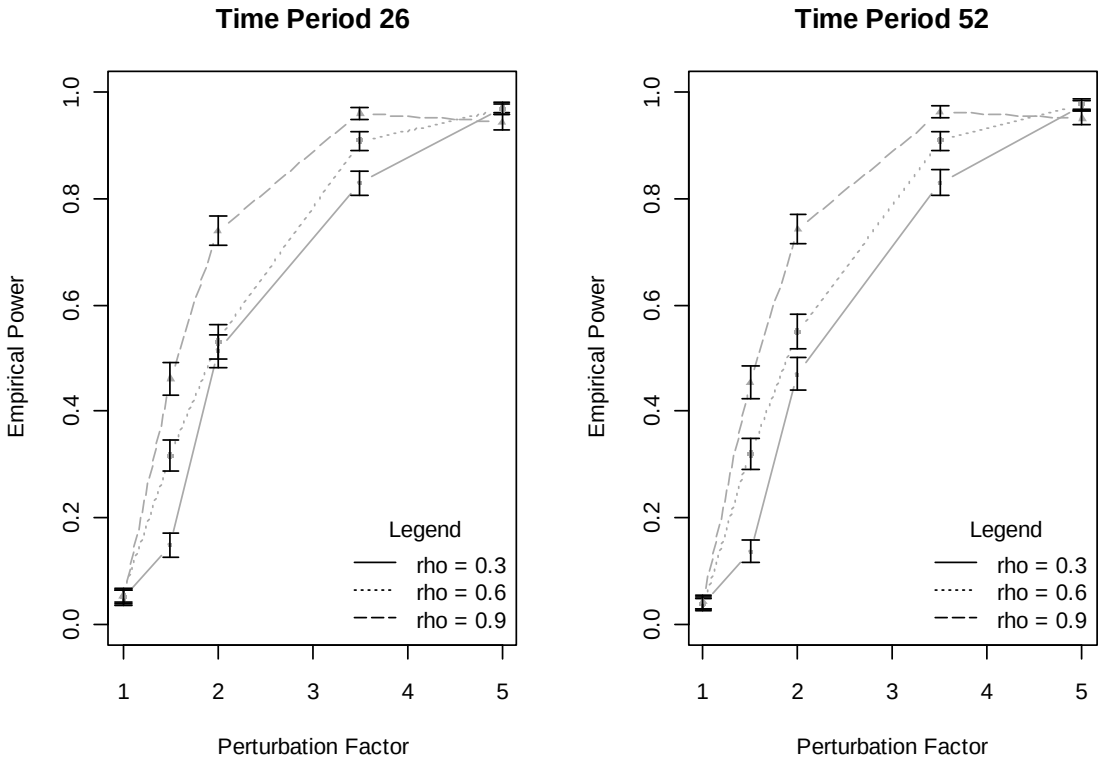
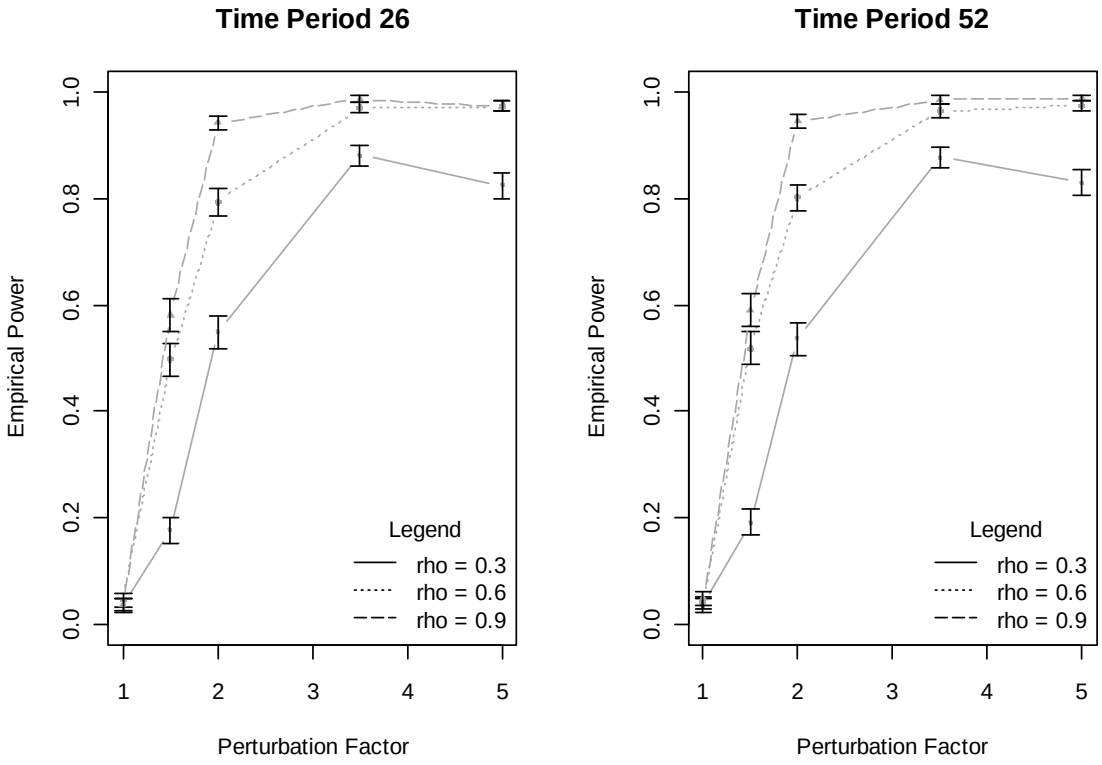


Figure 3-18: Plots of empirical power vs. perturbation factor for edge density = 0.9 and five edge perturbations; 2D configuration.



Inspection of figures 3-10 to 3-18 shows that for constant AR(1) correlation parameter, power increases as the perturbation factor increases. One exception to this can be found in figure 3-18; power for an AR(1) correlation of 0.3, the magnitude of power is higher for a perturbation factor of 3.5 versus the power for a perturbation factor of 5. However, the confidence intervals for power overlap for these two cases, and thus are statistically tied.

3.2.1.3 Power versus AR(1) Correlation Parameter

Figure 3-19: Plots of empirical power vs. AR(1) correlation for edge density = 0.3 and a single edge perturbation; 2D configuration.

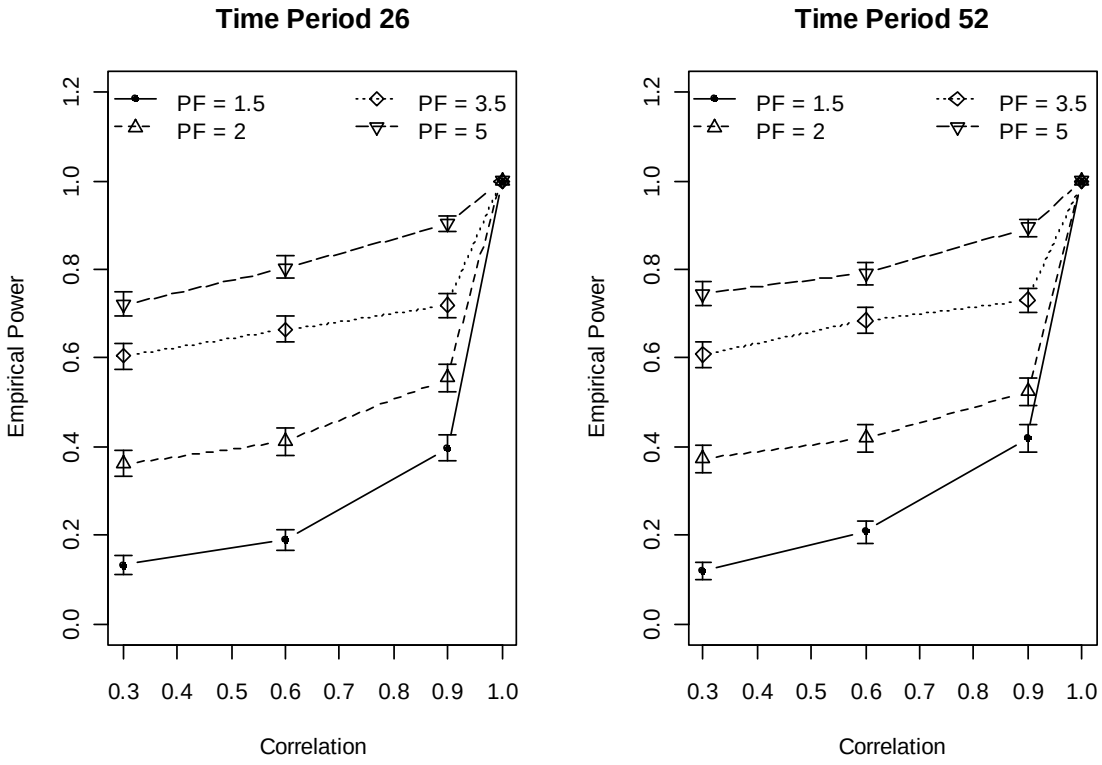


Figure 3-20: Plots of empirical power vs. AR(1) correlation for edge density = 0.3 and three edge perturbations; 2D configuration.

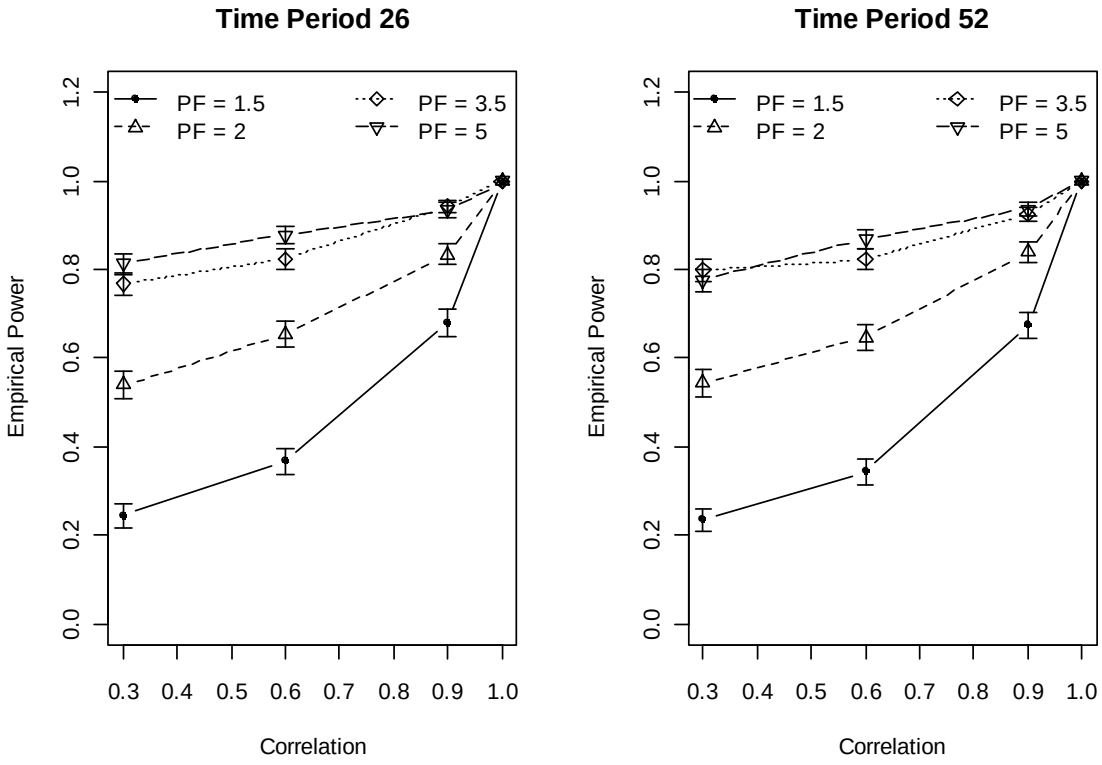


Figure 3-21: Plots of empirical power vs. AR(1) correlation for edge density = 0.3 and five edge perturbations; 2D configuration.

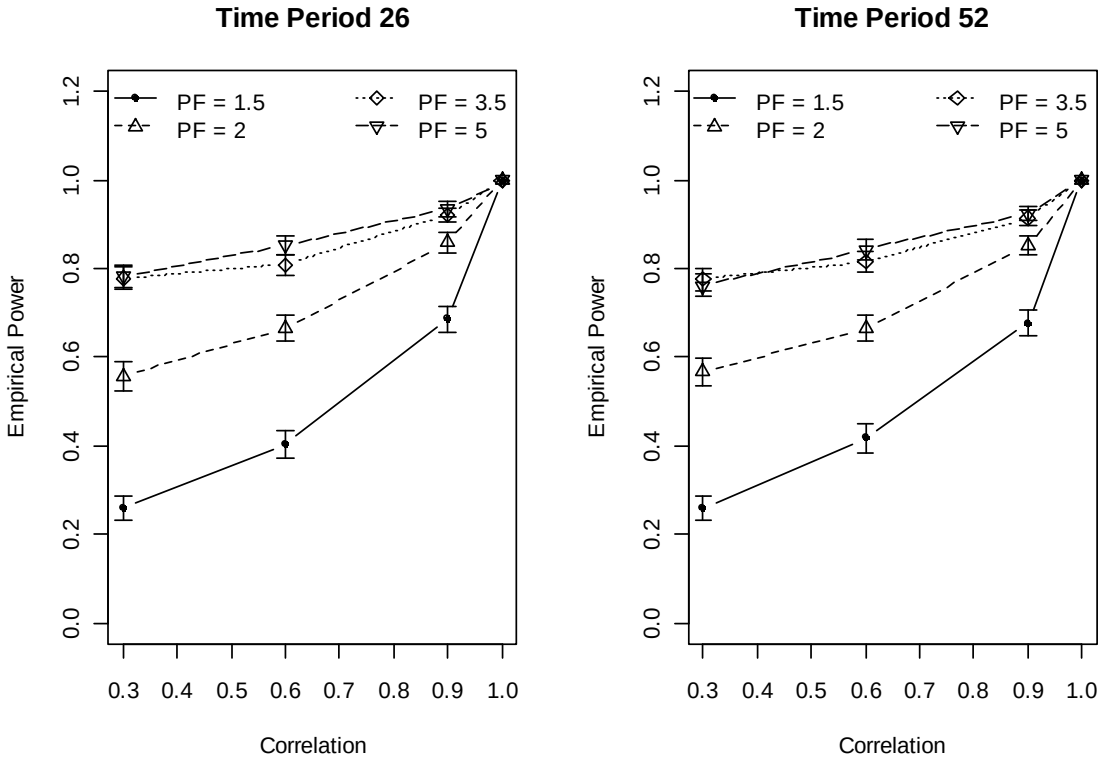


Figure 3-22: Plots of empirical power vs. AR(1) correlation for edge density = 0.6 and a single edge perturbation; 2D configuration.

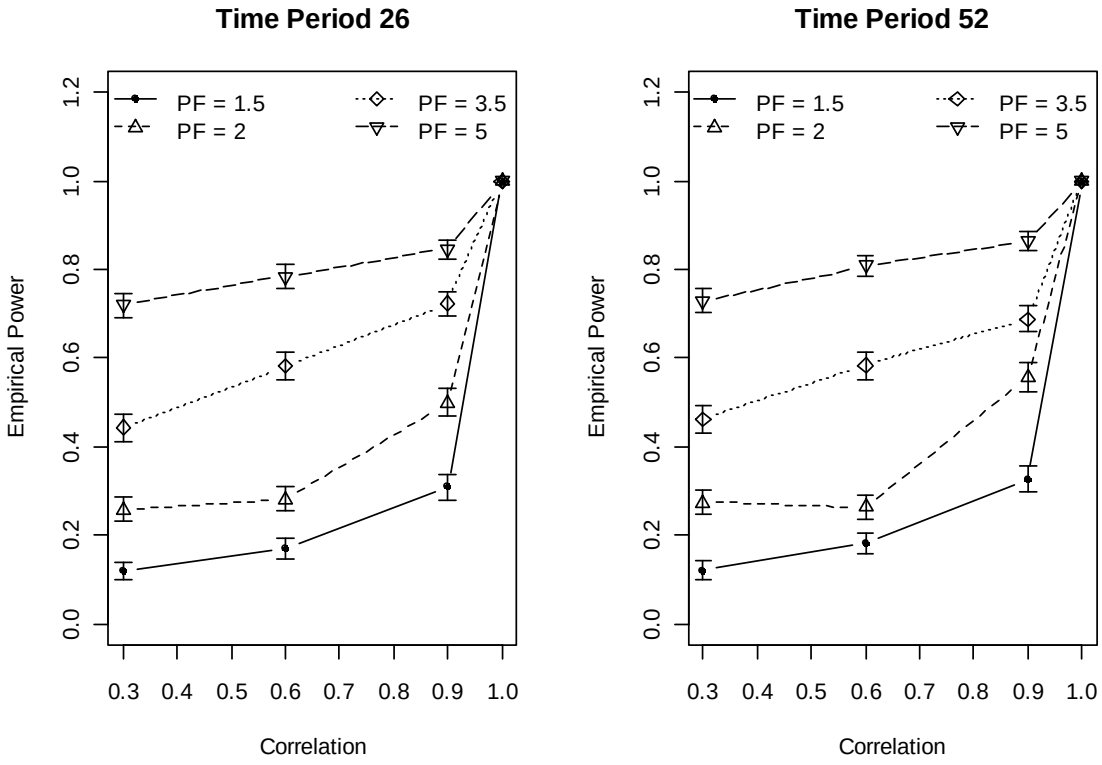


Figure 3-23: Plots of empirical power vs. AR(1) correlation for edge density = 0.6 and three edge perturbations; 2D configuration.

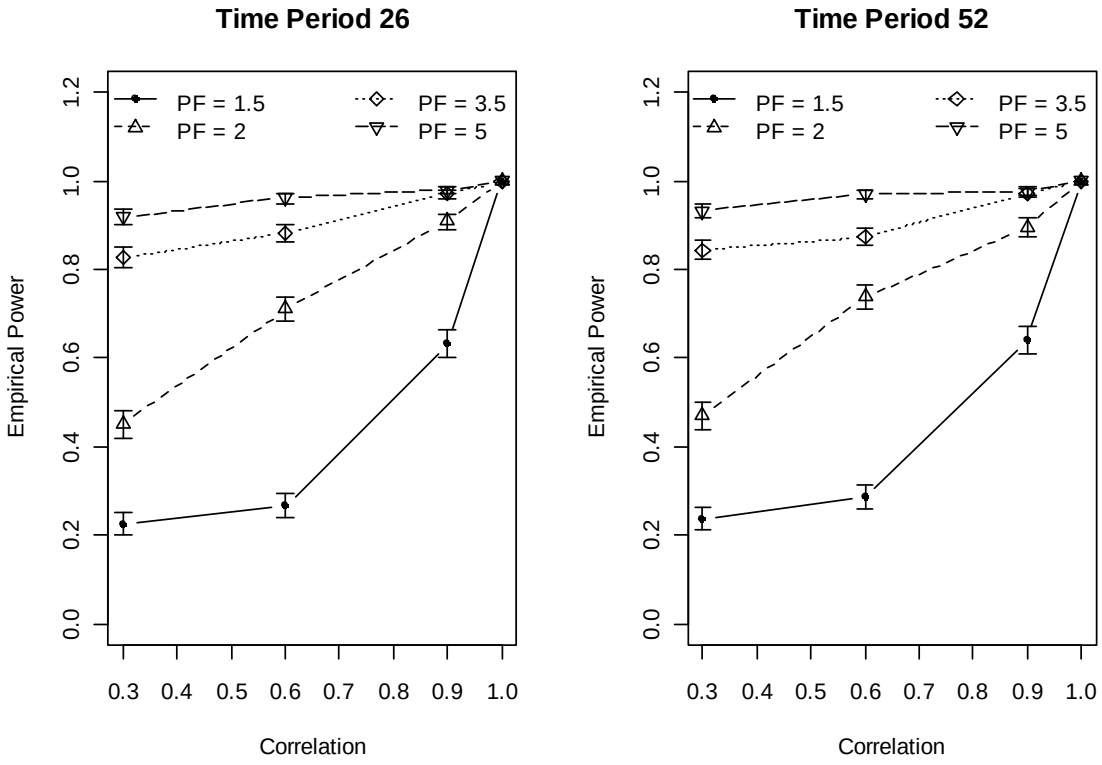


Figure 3-24: Plots of empirical power vs. AR(1) correlation for edge density = 0.6 and five edge perturbations; 2D configuration.

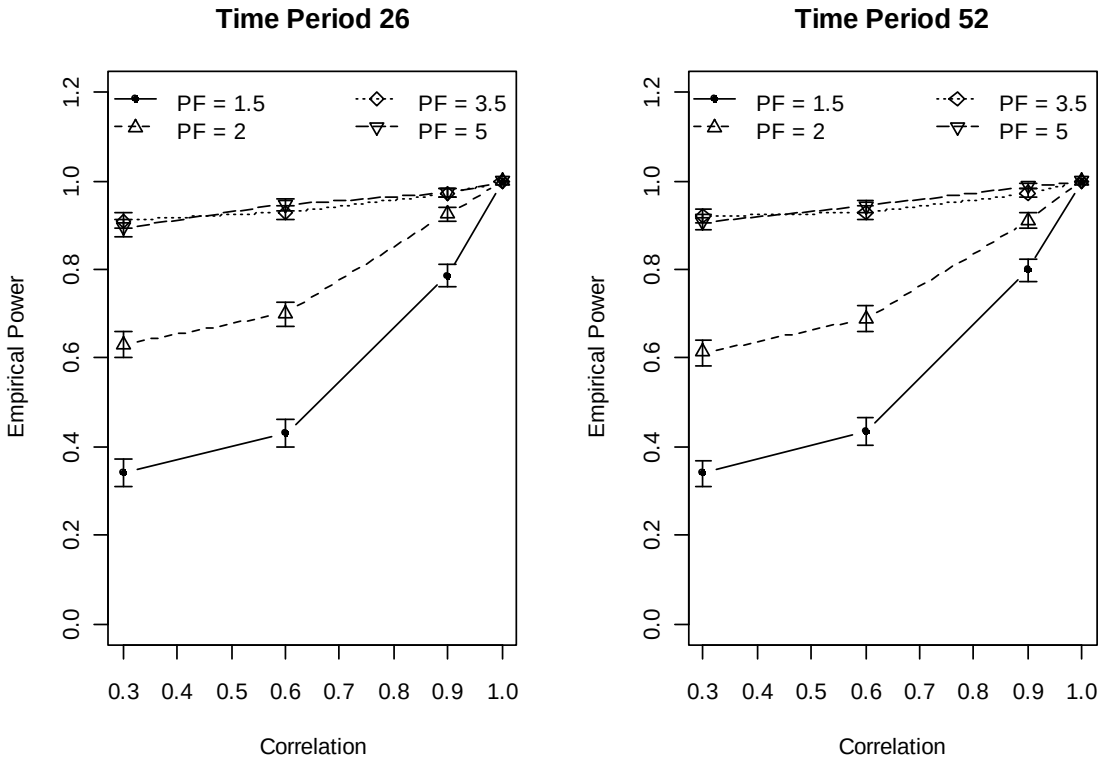


Figure 3-25: Plots of empirical power vs. AR(1) correlation for edge density = 0.9 and a single edge perturbation; 2D configuration.

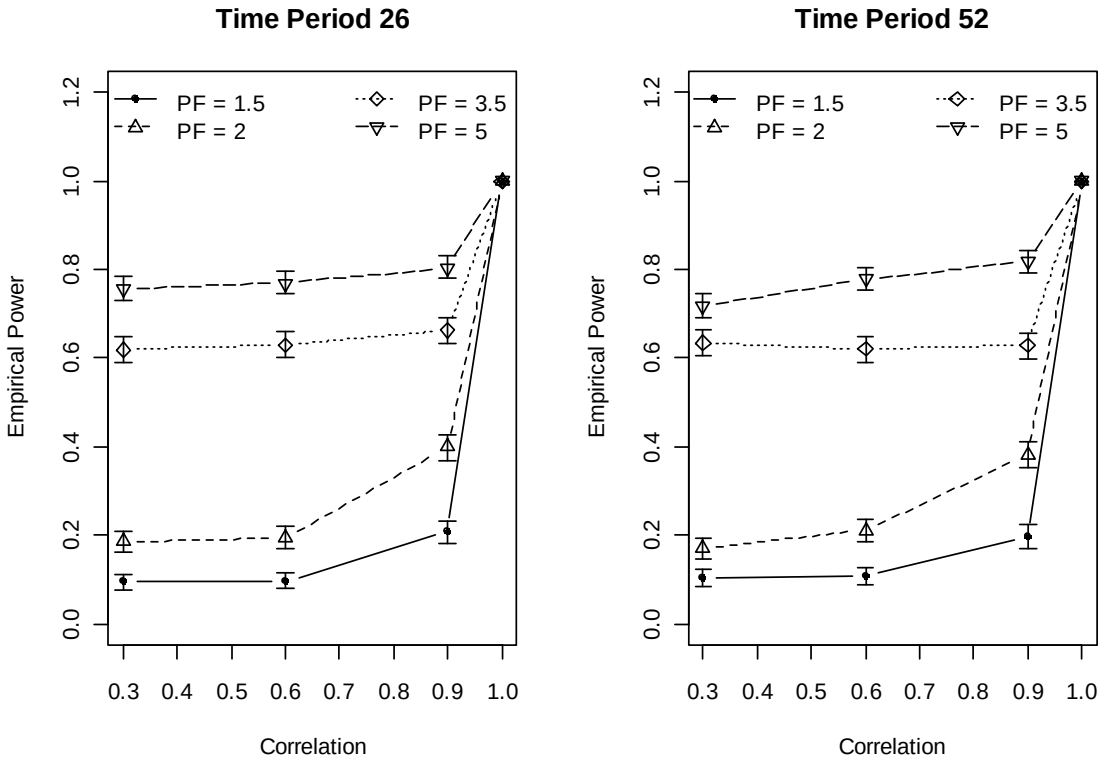


Figure 3-26: Plots of empirical power vs. AR(1) correlation for edge density = 0.9 and a three edge perturbation; 2D configuration.

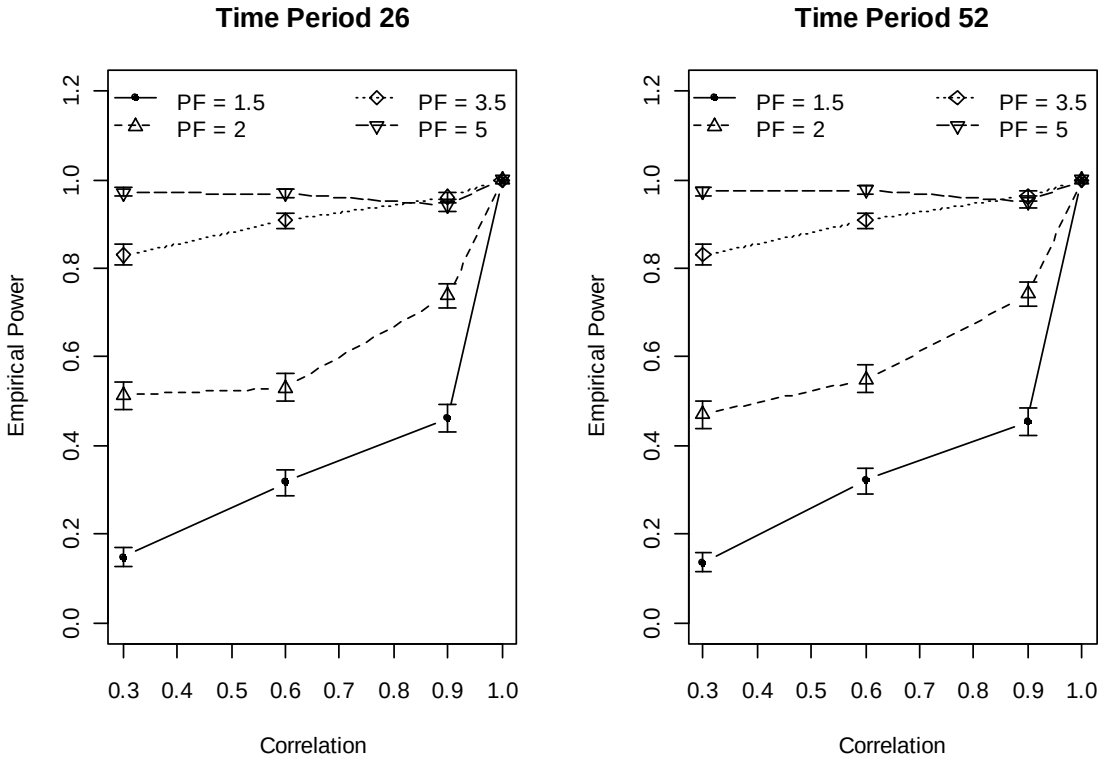
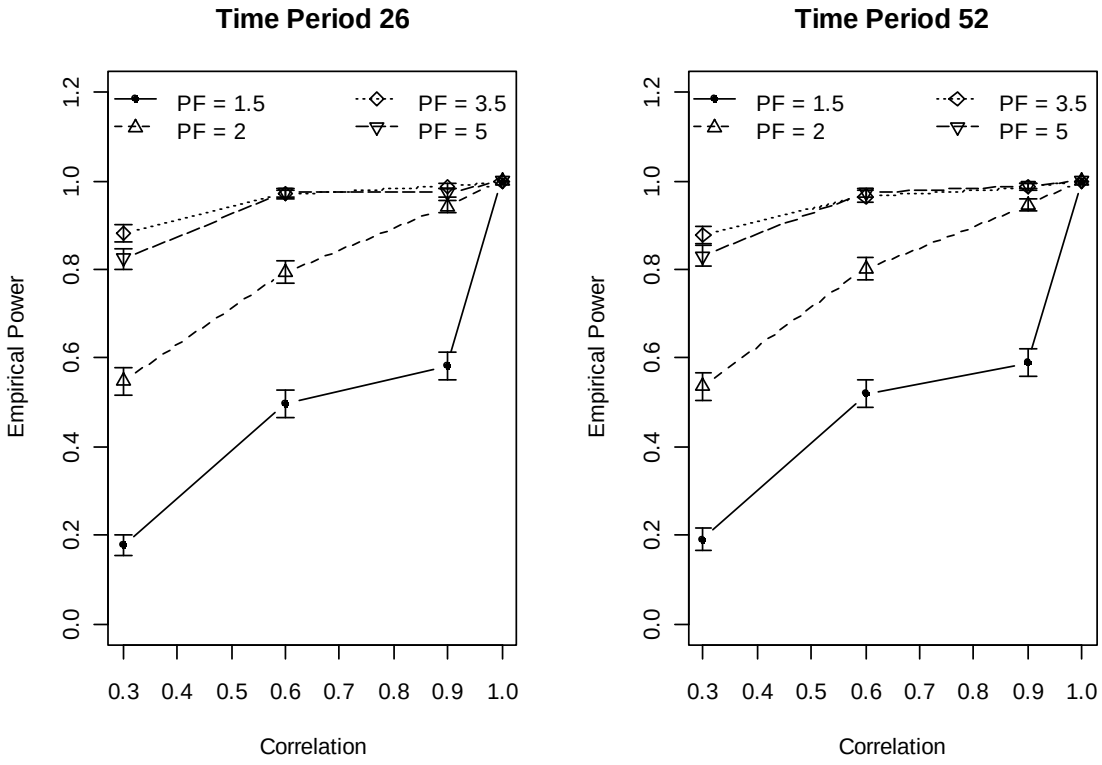


Figure 3-27: Plots of empirical power vs. AR(1) correlation for edge density = 0.9 and a five edge perturbation; 2D configuration.



Inspection of figures 3-19 to 3-27 reveals that power remains statistically constant (adjacent confidence intervals overlap) or increases as the AR(1) correlation parameter increases. The difference in power between lines of constant perturbation factor all decrease as the AR(1) correlation parameter increases, culminating in all combinations of factors yielding a power of 1 when the AR(1) correlation is 1.

3.2.2Empirical Power for Three Dimensional Configurations

Table 3-5: Type I error for tests based on 3D configurations.

Time of Perturbation	Number of Edges Perturbed	Edge Density	AR(1) Correlation Parameter					
			0.3		0.6		0.9	
			Type I Error	Std. Error	Type I Error	Std. Error	Type I Error	Std. Error
26 ($\alpha = 0.042$)	1	0.3	0.031	0.005	0.037	0.006	0.036	0.006
		0.6	0.052	0.007	0.037	0.006	0.052	0.007
		0.9	0.049	0.007	0.037	0.006	0.059	0.007
	3	0.3	0.049	0.008	0.033	0.006	0.030	0.005
		0.6	0.044	0.006	0.038	0.006	0.045	0.007
		0.9	0.045	0.007	0.050	0.007	0.047	0.007
	5	0.3	0.039	0.006	0.042	0.006	0.033	0.006
		0.6	0.048	0.007	0.047	0.007	0.055	0.007
		0.9	0.045	0.007	0.045	0.007	0.045	0.007
52 ($\alpha = 0.04$)	1	0.3	0.044	0.006	0.031	0.005	0.029	0.005
		0.6	0.042	0.006	0.045	0.007	0.045	0.007
		0.9	0.043	0.006	0.037	0.006	0.052	0.007
	3	0.3	0.037	0.006	0.037	0.006	0.033	0.006
		0.6	0.039	0.006	0.041	0.006	0.034	0.006
		0.9	0.049	0.007	0.034	0.006	0.036	0.006
	5	0.3	0.035	0.006	0.038	0.006	0.041	0.006
		0.6	0.036	0.006	0.051	0.007	0.039	0.006
		0.9	0.031	0.005	0.037	0.006	0.049	0.007

Table 3-6: Power for one edge perturbation at each combination of edge density, AR(1) correlation parameter, perturbation factor (PF) and time period of perturbation (TP).

TP	PF	Edge Density	AR(1) Correlation Parameter							
			0.3		0.6		0.9		1.0	
			Power	Std. Error	Power	Std. Error	Power	Std. Error	Power	Std. Error
26	1.5	0.3	0.127	0.011	0.204	0.013	0.420	0.016	1.000	-
		0.6	0.119	0.010	0.166	0.012	0.355	0.015	1.000	-
		0.9	0.083	0.009	0.126	0.010	0.295	0.014	1.000	-
	2	0.3	0.401	0.015	0.447	0.016	0.672	0.015	1.000	-
		0.6	0.281	0.014	0.361	0.015	0.618	0.015	1.000	-
		0.9	0.228	0.013	0.388	0.015	0.396	0.015	1.000	-
	3.5	0.3	0.687	0.015	0.743	0.014	0.807	0.012	1.000	-
		0.6	0.593	0.016	0.637	0.015	0.827	0.012	1.000	-
		0.9	0.650	0.015	0.624	0.015	0.735	0.014	1.000	-
	5	0.3	0.753	0.014	0.865	0.011	0.950	0.007	1.000	-
		0.6	0.760	0.014	0.856	0.011	0.865	0.011	1.000	-
		0.9	0.807	0.012	0.802	0.013	0.804	0.013	1.000	-
52	1.5	0.3	0.135	0.011	0.226	0.013	0.459	0.016	1.000	-
		0.6	0.122	0.010	0.180	0.012	0.361	0.015	1.000	-
		0.9	0.098	0.009	0.109	0.010	0.309	0.015	1.000	-
	2	0.3	0.409	0.016	0.458	0.016	0.633	0.015	1.000	-
		0.6	0.298	0.014	0.346	0.015	0.647	0.015	1.000	-
		0.9	0.237	0.013	0.405	0.016	0.406	0.016	1.000	-
	3.5	0.3	0.711	0.014	0.799	0.013	0.794	0.013	1.000	-
		0.6	0.608	0.015	0.671	0.015	0.798	0.013	1.000	-
		0.9	0.650	0.015	0.605	0.015	0.735	0.014	1.000	-
	5	0.3	0.766	0.013	0.868	0.011	0.936	0.008	1.000	-
		0.6	0.759	0.014	0.858	0.011	0.877	0.010	1.000	-
		0.9	0.771	0.013	0.796	0.013	0.811	0.012	1.000	-

Table 3-7: Power for three edge perturbations at each combination of edge density, AR(1) correlation parameter, perturbation factor (PF) and time period of perturbation (TP).

TP	PF	Edge Density	AR(1) Correlation Parameter							
			0.3		0.6		0.9		1.0	
			Power	Std. Error	Power	Std. Error	Power	Std. Error	Power	Std. Error
26	1.5	0.3	0.248	0.014	0.374	0.015	0.706	0.014	1.000	-
		0.6	0.257	0.014	0.282	0.014	0.677	0.015	1.000	-
		0.9	0.259	0.014	0.220	0.013	0.538	0.016	1.000	-
	2	0.3	0.606	0.015	0.689	0.015	0.900	0.009	1.000	-
		0.6	0.519	0.016	0.723	0.014	0.885	0.010	1.000	-
		0.9	0.495	0.016	0.620	0.015	0.761	0.013	1.000	-
	3.5	0.3	0.801	0.013	0.888	0.010	0.947	0.007	1.000	-
		0.6	0.894	0.010	0.956	0.006	0.984	0.004	1.000	-
		0.9	0.823	0.012	0.876	0.010	0.958	0.006	1.000	-
	5	0.3	0.827	0.012	0.891	0.010	0.960	0.006	1.000	-
		0.6	0.939	0.008	0.957	0.006	0.984	0.004	1.000	-
		0.9	0.937	0.008	0.982	0.004	0.976	0.005	1.000	-
52	1.5	0.3	0.267	0.014	0.393	0.015	0.681	0.015	1.000	-
		0.6	0.240	0.014	0.320	0.015	0.688	0.015	1.000	-
		0.9	0.283	0.014	0.216	0.013	0.512	0.016	1.000	-
	2	0.3	0.618	0.015	0.719	0.014	0.898	0.010	1.000	-
		0.6	0.542	0.016	0.731	0.014	0.878	0.010	1.000	-
		0.9	0.536	0.016	0.670	0.015	0.760	0.014	1.000	-
	3.5	0.3	0.849	0.011	0.905	0.009	0.955	0.007	1.000	-
		0.6	0.892	0.010	0.949	0.007	0.989	0.003	1.000	-
		0.9	0.845	0.011	0.894	0.010	0.956	0.006	1.000	-
	5	0.3	0.807	0.012	0.903	0.009	0.962	0.006	1.000	-
		0.6	0.925	0.008	0.959	0.006	0.980	0.004	1.000	-
		0.9	0.944	0.007	0.982	0.004	0.977	0.005	1.000	-

Table 3-8: Power for five edge perturbations at each combination of edge density, AR(1) correlation parameter, perturbation factor (PF) and time period of perturbation (TP).

TP	PF	Edge Density	AR(1) Correlation Parameter							
			0.3		0.6		0.9		1.0	
			Power	Std. Error	Power	Std. Error	Power	Std. Error	Power	Std. Error
26	1.5	0.3	0.282	0.014	0.444	0.016	0.764	0.013	1.000	-
		0.6	0.335	0.015	0.468	0.016	0.819	0.012	1.000	-
		0.9	0.365	0.015	0.318	0.015	0.841	0.012	1.000	-
	2	0.3	0.598	0.016	0.774	0.013	0.904	0.009	1.000	-
		0.6	0.672	0.015	0.796	0.013	0.950	0.007	1.000	-
		0.9	0.629	0.015	0.698	0.015	0.937	0.008	1.000	-
	3.5	0.3	0.777	0.013	0.871	0.011	0.956	0.006	1.000	-
		0.6	0.923	0.008	0.954	0.007	0.984	0.004	1.000	-
		0.9	0.926	0.008	0.964	0.006	0.992	0.003	1.000	-
	5	0.3	0.836	0.012	0.892	0.010	0.959	0.006	1.000	-
		0.6	0.939	0.008	0.968	0.006	0.985	0.004	1.000	-
		0.9	0.921	0.009	0.942	0.007	0.988	0.003	1.000	-
52	1.5	0.3	0.292	0.014	0.431	0.016	0.773	0.013	1.000	-
		0.6	0.340	0.016	0.455	0.015	0.821	0.012	1.000	-
		0.9	0.360	0.015	0.313	0.015	0.842	0.012	1.000	-
	2	0.3	0.638	0.015	0.735	0.014	0.915	0.009	1.000	-
		0.6	0.699	0.015	0.791	0.013	0.947	0.007	1.000	-
		0.9	0.627	0.015	0.683	0.015	0.925	0.008	1.000	-
	3.5	0.3	0.785	0.013	0.882	0.010	0.949	0.007	1.000	-
		0.6	0.933	0.008	0.964	0.006	0.987	0.004	1.000	-
		0.9	0.933	0.008	0.969	0.005	0.995	0.002	1.000	-
	5	0.3	0.860	0.011	0.901	0.009	0.963	0.006	1.000	-
		0.6	0.945	0.007	0.961	0.006	0.995	0.002	1.000	-
		0.9	0.910	0.009	0.948	0.007	0.993	0.003	1.000	-

3.2.2.1 Power versus Number of Perturbed Edges

Figure 3-28: Plots of empirical power vs. number of edges perturbed for edge density = 0.3 and AR(1) correlation parameter = 0.3; 3D configuration.

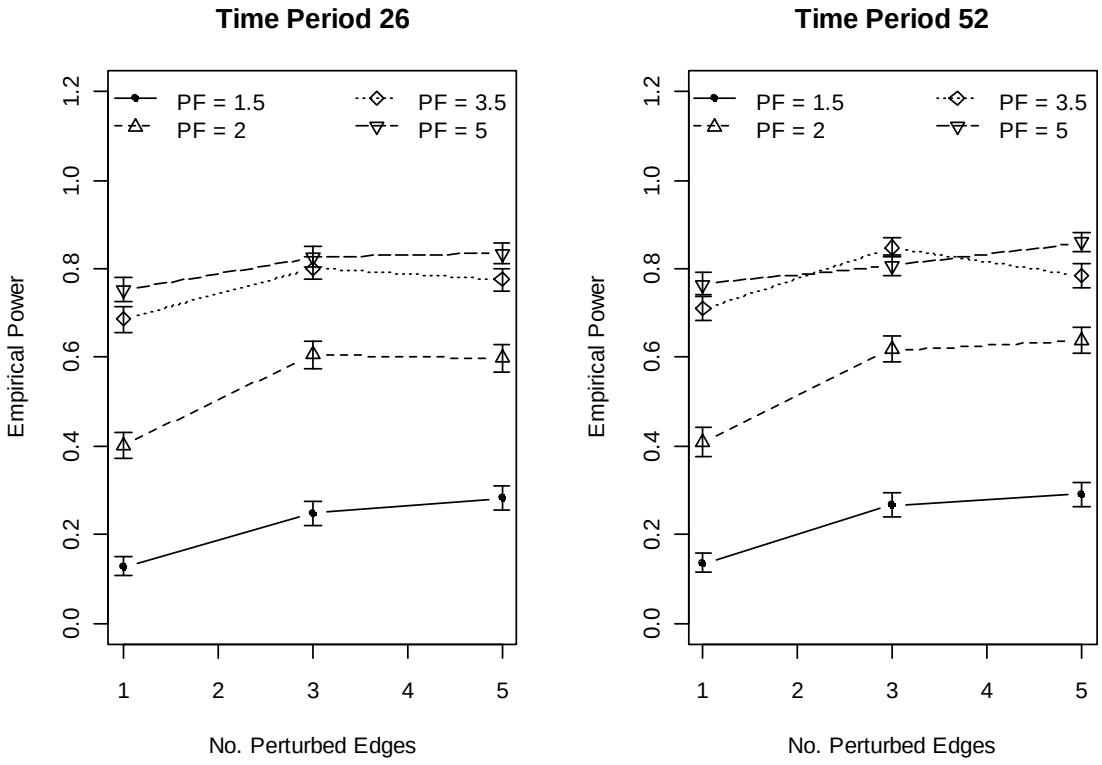


Figure 3-29: Plots of empirical power vs. number of edges perturbed for edge density = 0.3 and AR(1) correlation parameter = 0.6; 3D configuration.

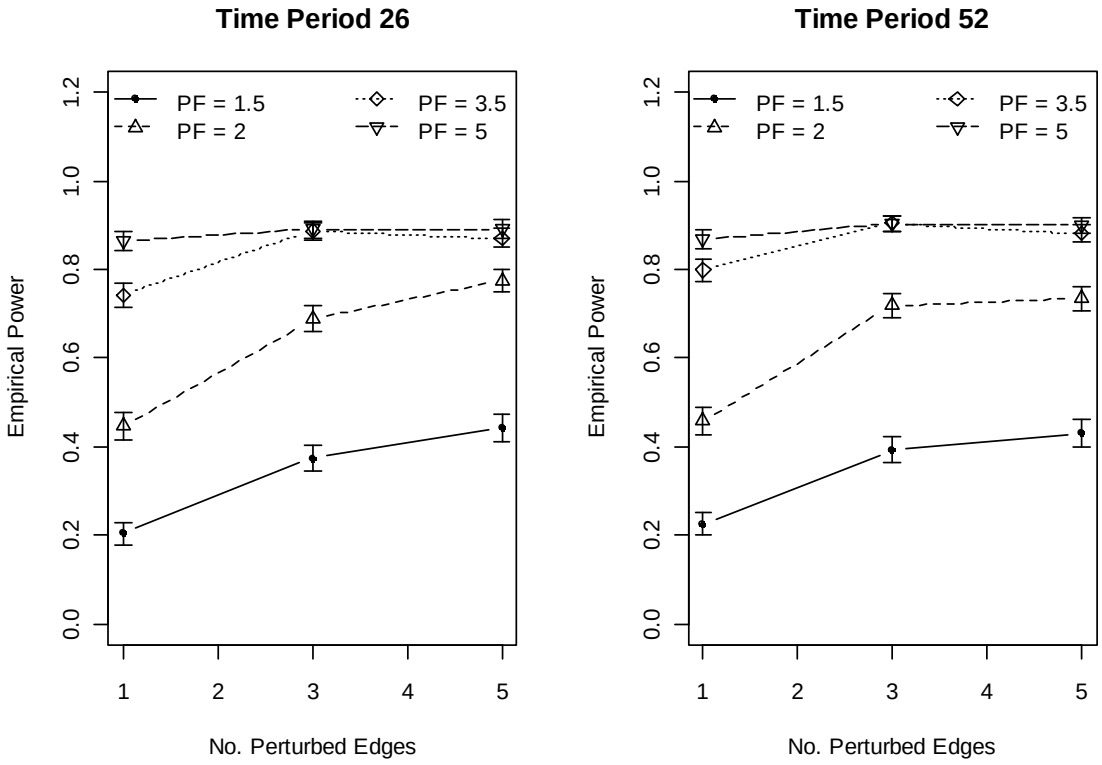


Figure 3-30: Plots of empirical power vs. number of edges perturbed for edge density = 0.3 and AR(1) correlation parameter = 0.9; 3D configuration.

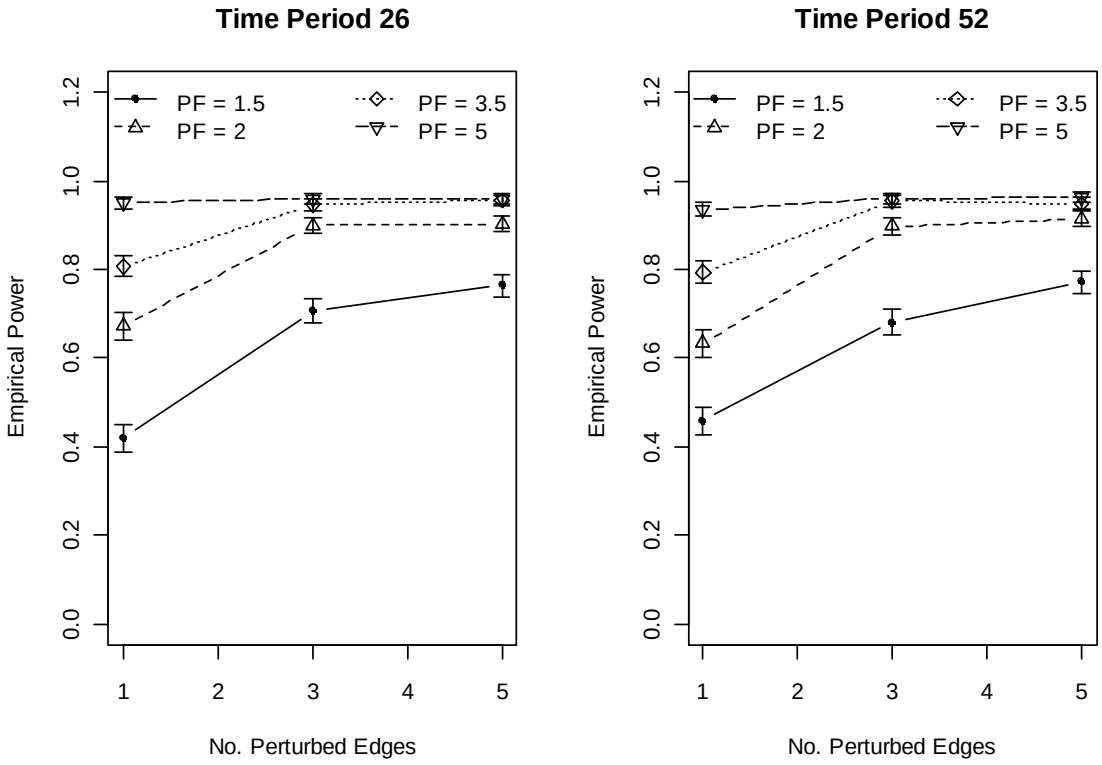


Figure 3-31: Plots of empirical power vs. number of edges perturbed for edge density = 0.6 and AR(1) correlation parameter = 0.3; 3D configuration.

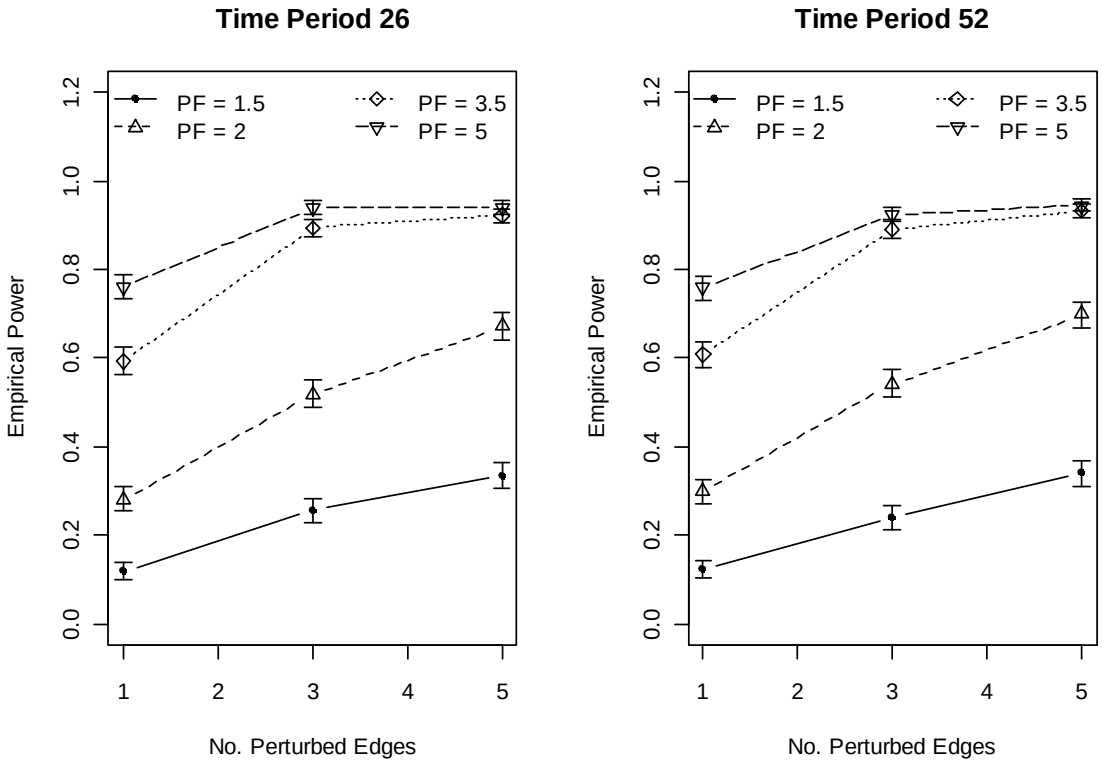


Figure 3-32: Plots of empirical power vs. number of edges perturbed for edge density = 0.6 and AR(1) correlation parameter = 0.6; 3D configuration.

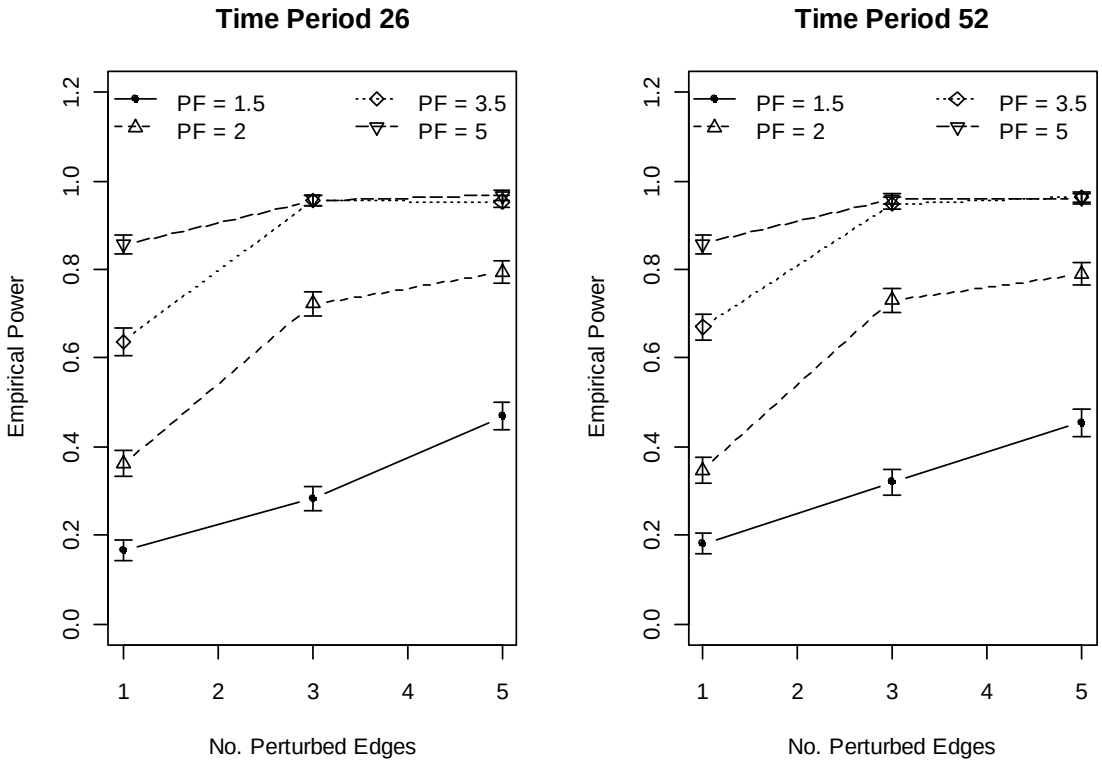


Figure 3-33: Plots of empirical power vs. number of edges perturbed for edge density = 0.6 and AR(1) correlation parameter = 0.6; 3D configuration.

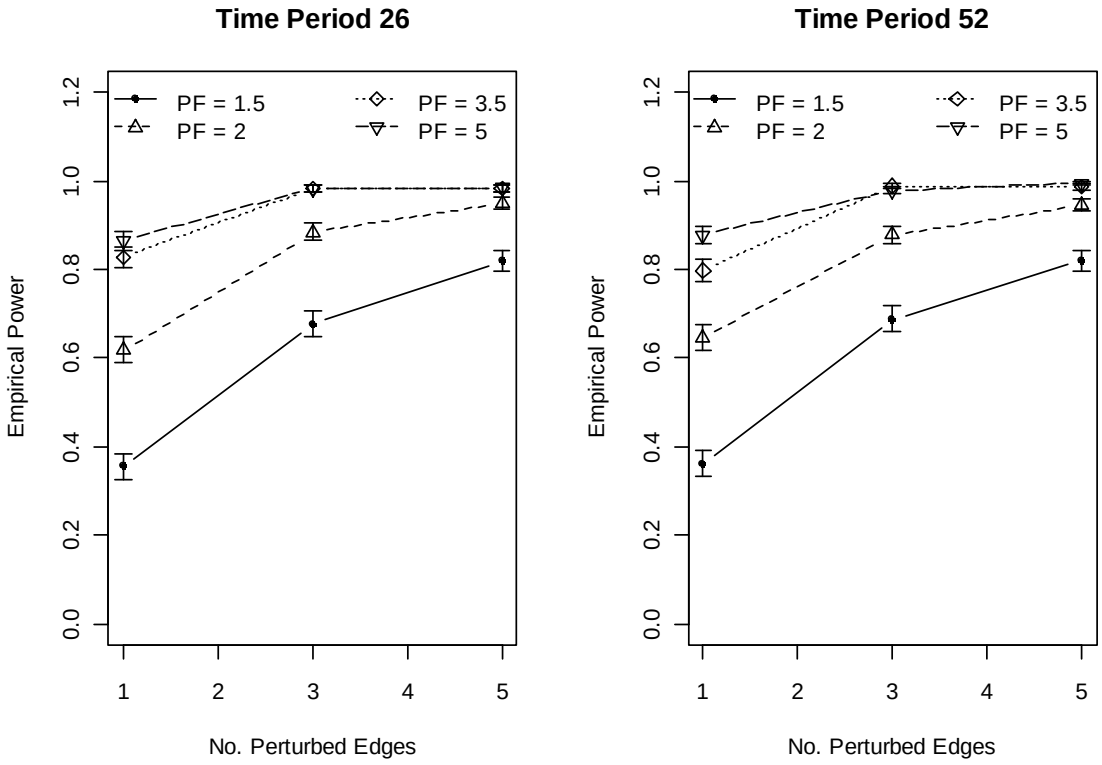


Figure 3-34: Plots of empirical power vs. number of edges perturbed for edge density = 0.9 and AR(1) correlation parameter = 0.3; 3D configuration.

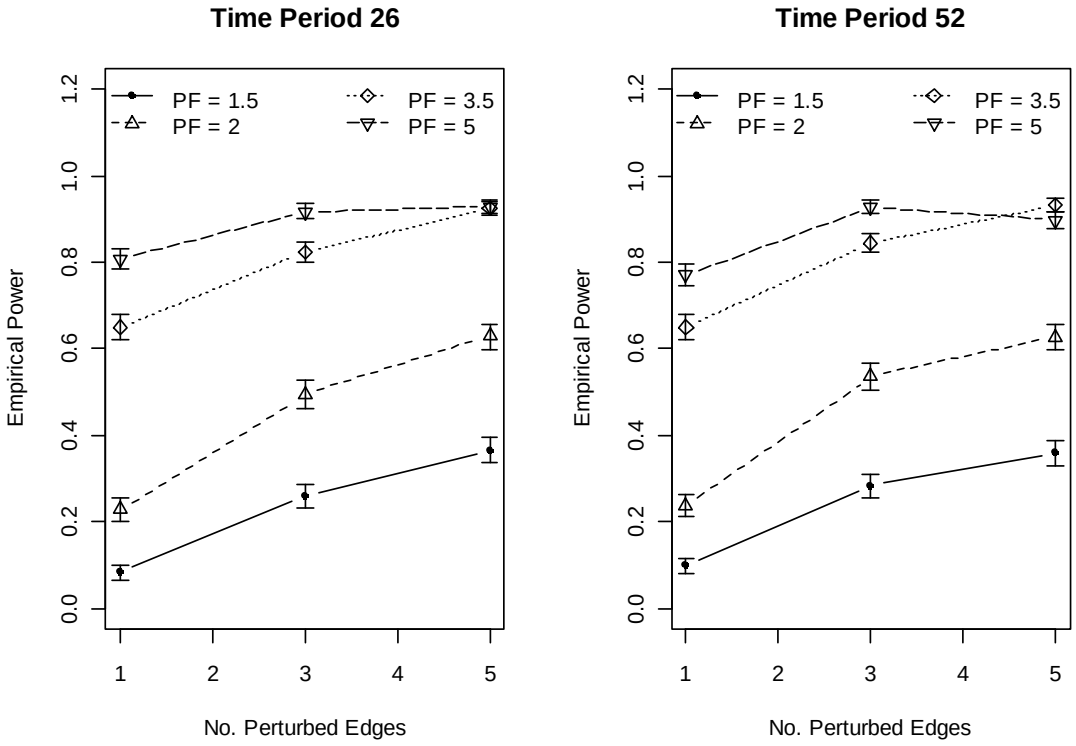


Figure 3-35: Plots of empirical power vs. number of edges perturbed for edge density = 0.9 and AR(1) correlation parameter = 0.6; 3D configuration.

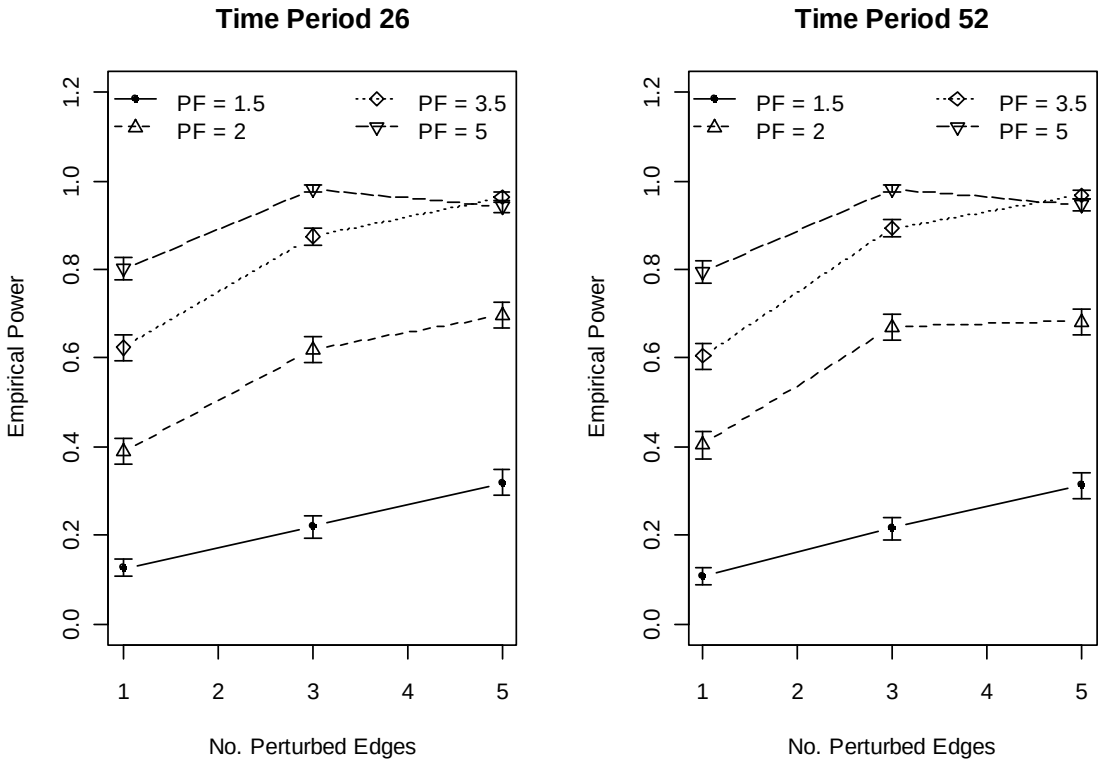
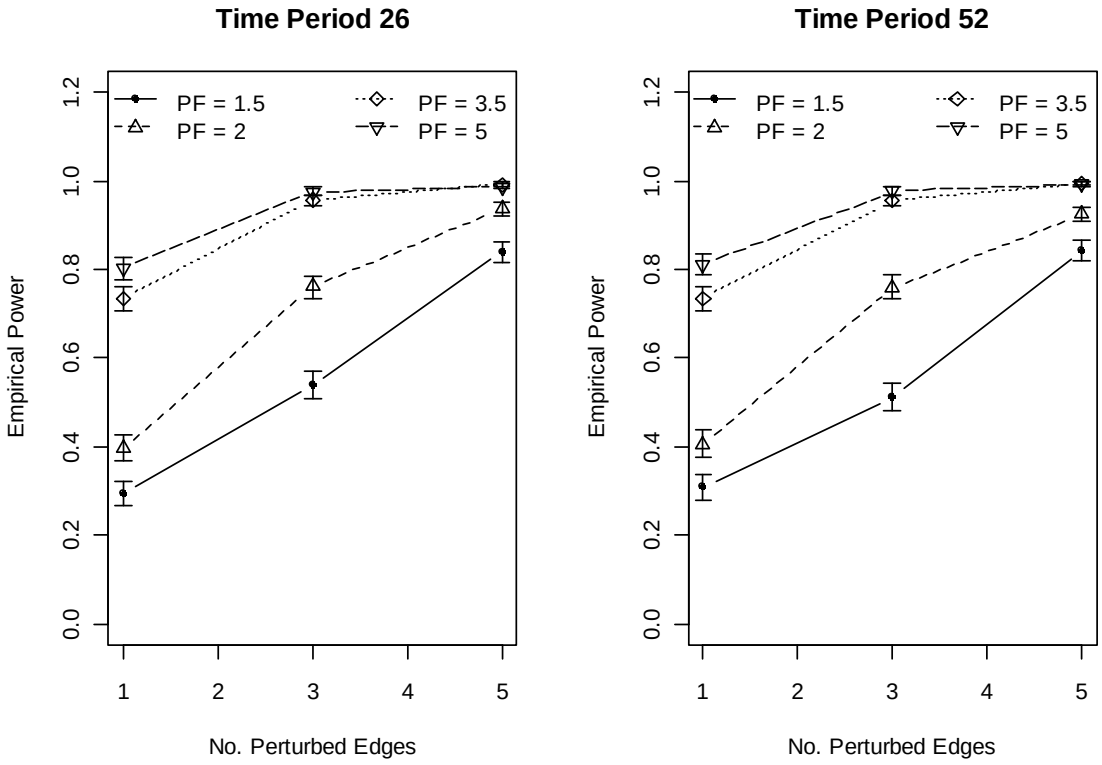


Figure 3-36: Plots of empirical power vs. number of edges perturbed for edge density = 0.9 and AR(1) correlation parameter = 0.9; 3D configuration.



Inspection of figures 3-28 to 3-36 reveals similar results to those for the 2D configurations. With the exception of time period 52 in figure 3-28 and both time periods in figure 3-35, the shape of the power versus number of perturbed edges, regardless of edge density, are very similar. Specifically, for a perturbation factor of 1.5, power remains significantly below the power for all other perturbation factors, with only few instances where power is at or above 0.060. Other than for single edge perturbations, the power for a perturbation factor of 2 is at or above 0.060 for all edge densities when the AR(1) correlation parameter is 0.6 or 0.9.

In figure 3-28, the lower bound for the confidence interval for power at an edge density of 0.3, AR(1) correlation parameter of 0.3, the 52nd time period and three edges perturbed with a perturbation factor of 3.5 is 0.827 while the upper bound for five edges at a perturbation factor of 5 is 0.810. In figure 3-35, the lower bound for the confidence interval for power at an edge density of 0.9, AR(1) correlation parameter of 0.6, the 26th (52nd) time period and three edges perturbed with a factor of 5 is 0.966 (0.975) while the upper bound for five edges at a perturbation factor of 5 is 0.956 (0.962). These three results may be a manifestation of the dipping or leveling off of the aR^2 with number of dimensions discussed at the end of chapter 2 rather than an indictment of the method. In the remaining figures, all combinations of perturbation factors 3.5 and 5 and perturbed edges 3 and 5, the confidence intervals for power overlap. Other than the three cases outlined above, power increased or stayed within the adjacent margin of error as the number of perturbed edges increased.

3.2.2.2 Power versus Perturbation Factor

Figure 3-37: Plots of empirical power vs. perturbation factor for edge density = 0.3 and a single edge perturbation; 3D configuration.

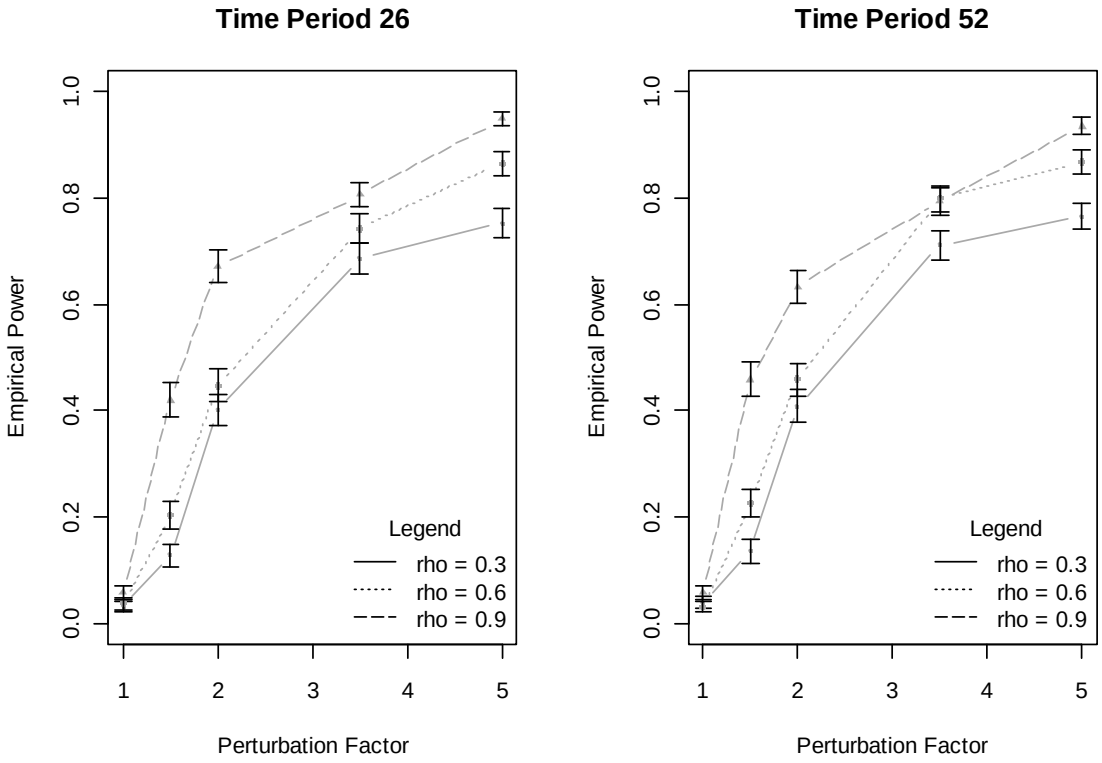


Figure 3-38: Plots of empirical power vs. perturbation factor for edge density = 0.3 and three edge perturbations; 3D configuration.

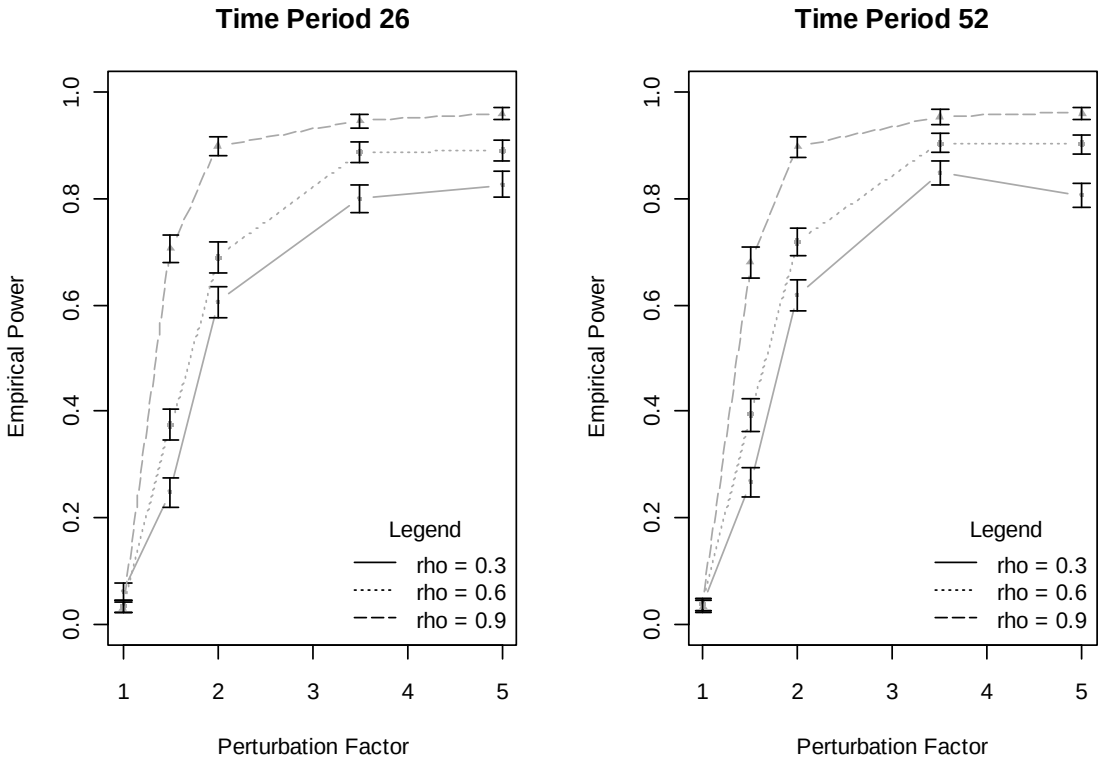


Figure 3-39: Plots of empirical power vs. perturbation factor for edge density = 0.3 and five edge perturbations; 3D configuration.

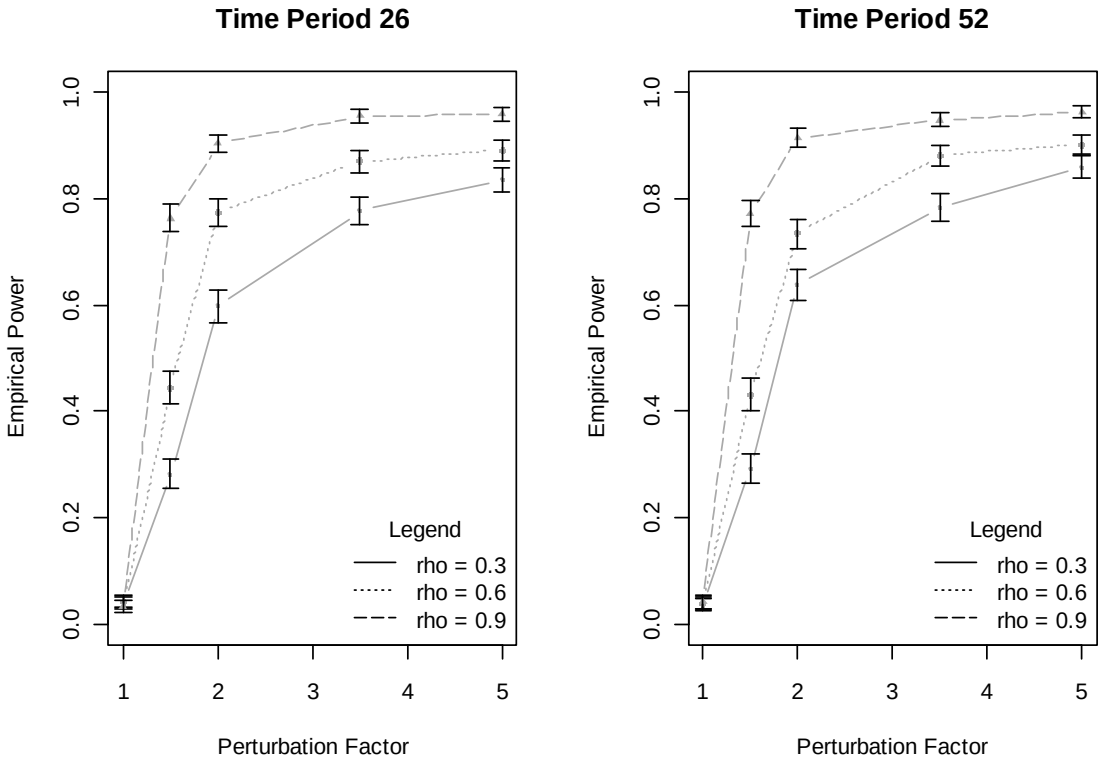


Figure 3-40: Plots of empirical power vs. perturbation factor for edge density = 0.6 and a single edge perturbation; 3D configuration.

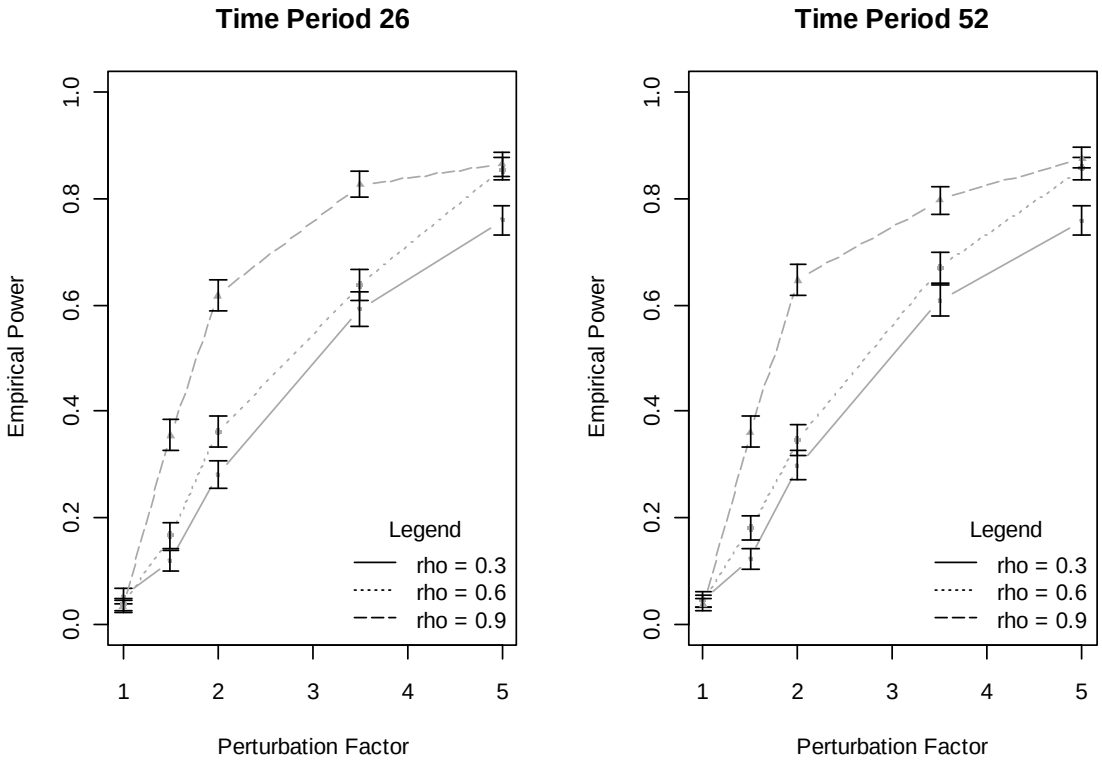


Figure 3-41: Plots of empirical power vs. perturbation factor for edge density = 0.6 and three edge perturbations; 3D configuration.

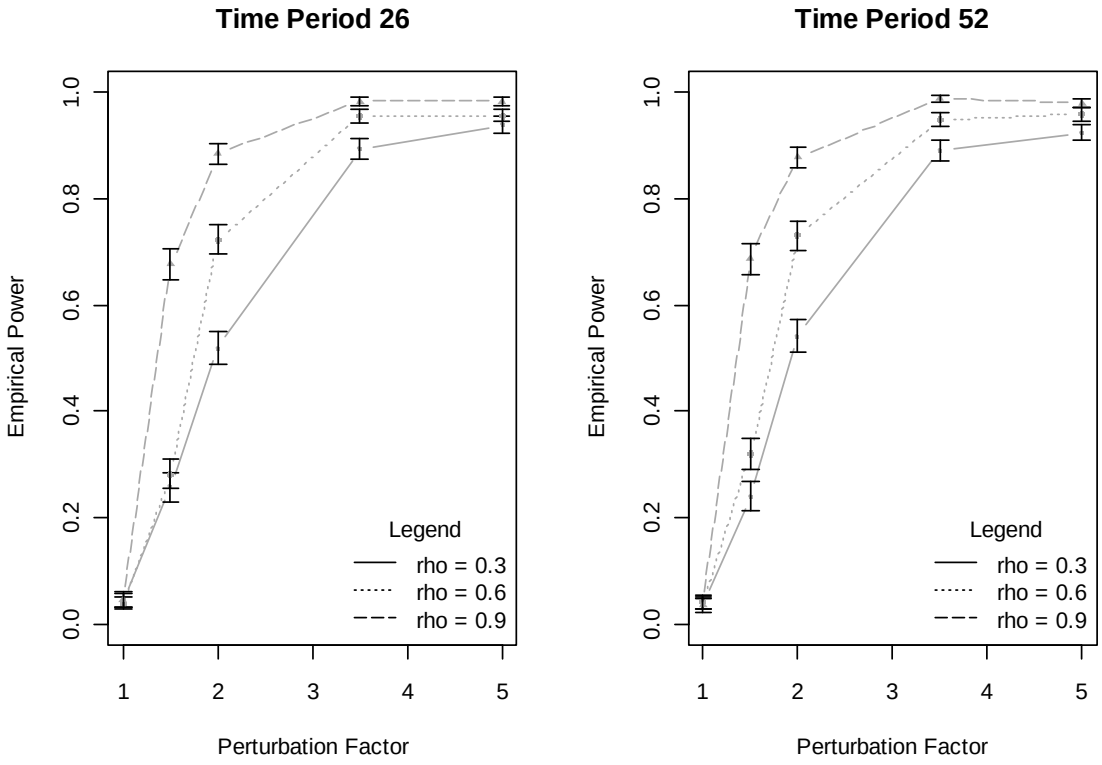


Figure 3-42: Plots of empirical power vs. perturbation factor for edge density = 0.6 and five edge perturbations; 3D configuration.

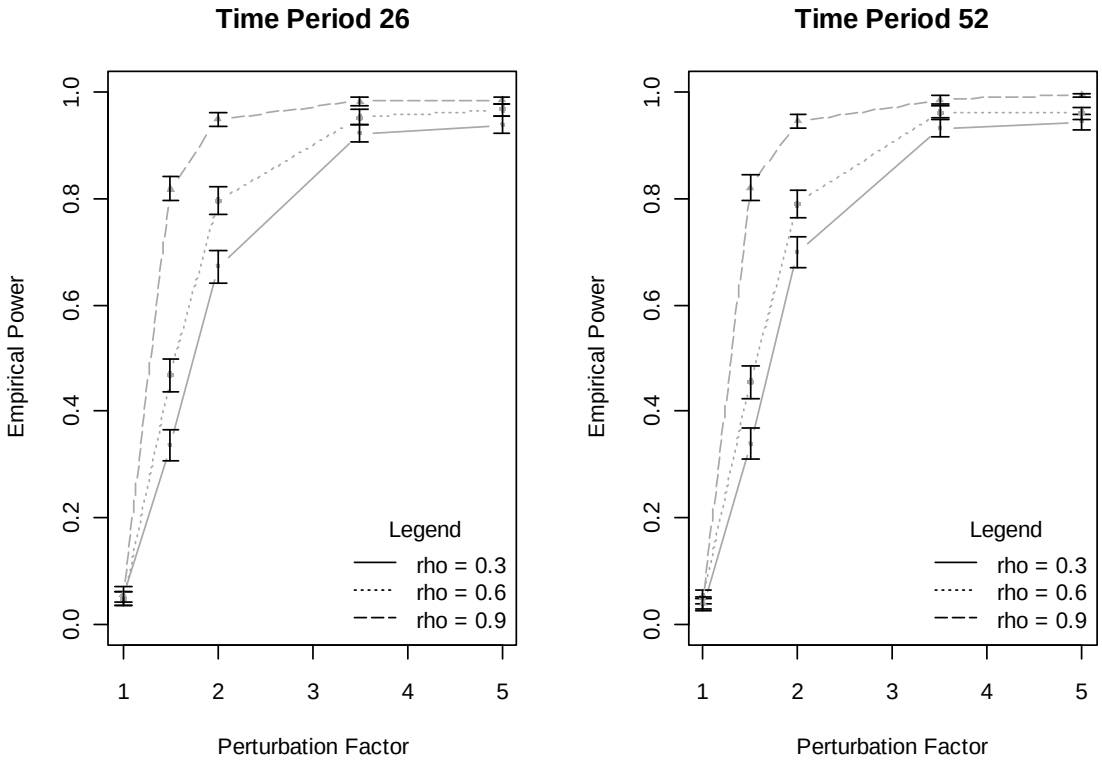


Figure 3-43: Plots of empirical power vs. perturbation factor for edge density = 0.9 and a single edge perturbation; 3D configuration.

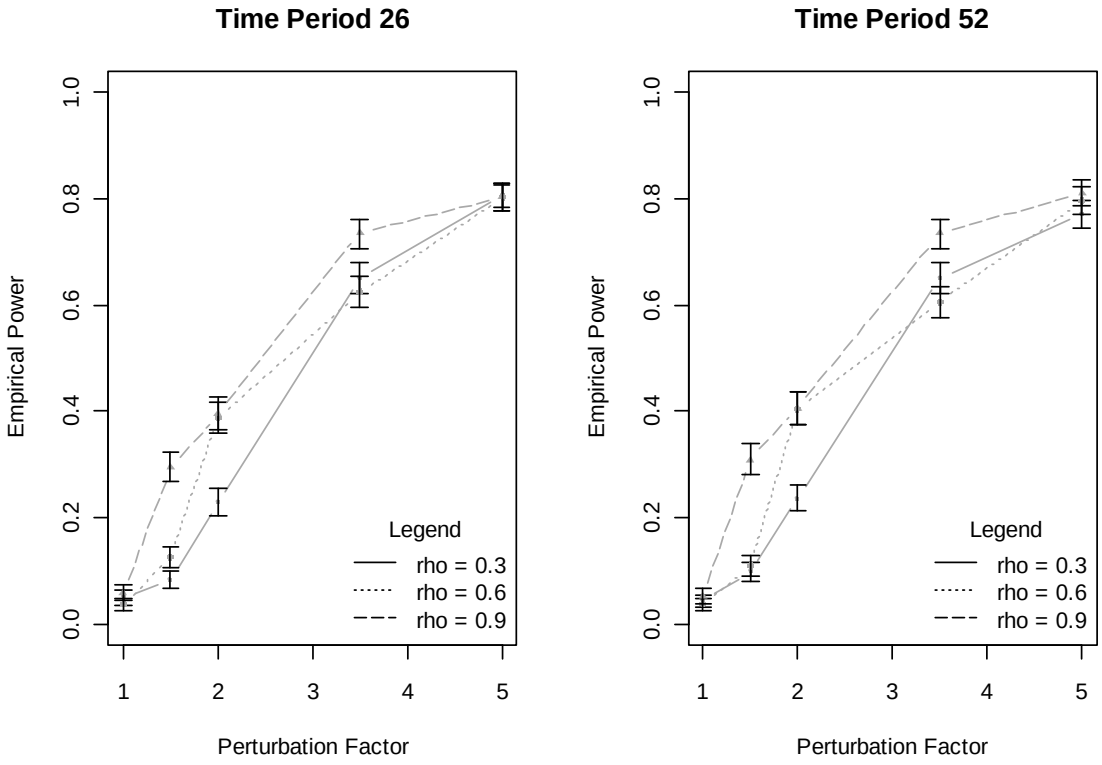


Figure 3-44: Plots of empirical power vs. perturbation factor for edge density = 0.9 and three edge perturbations; 3D configuration.

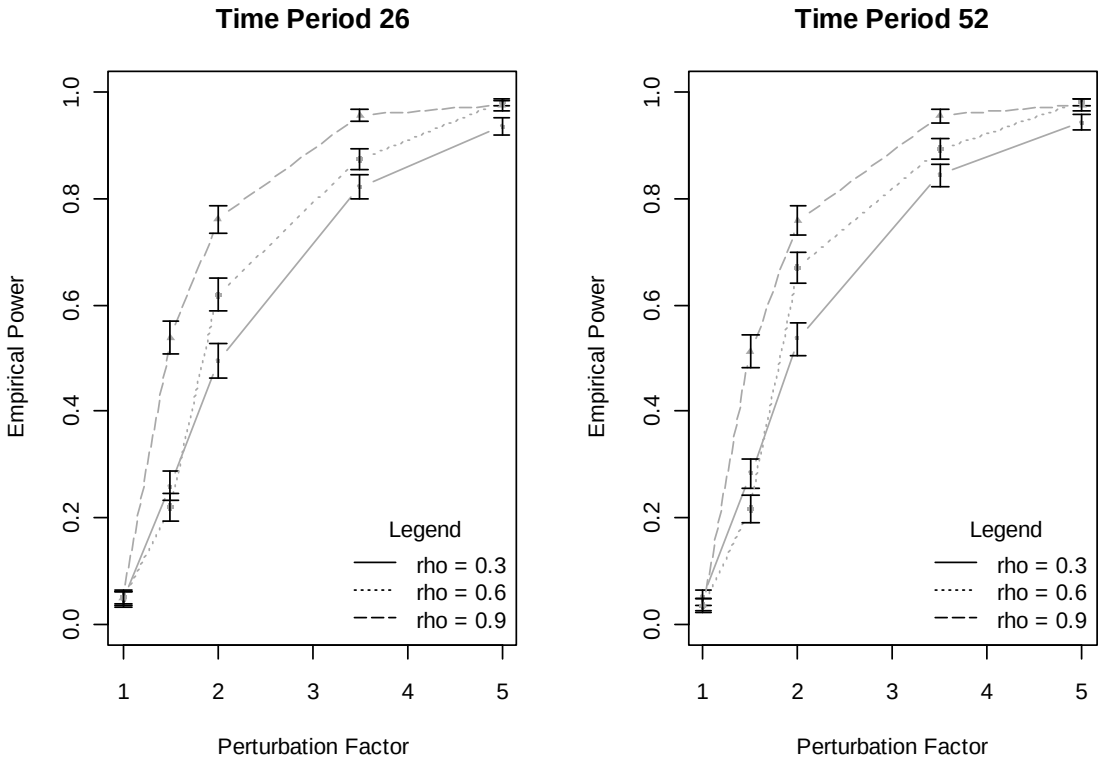
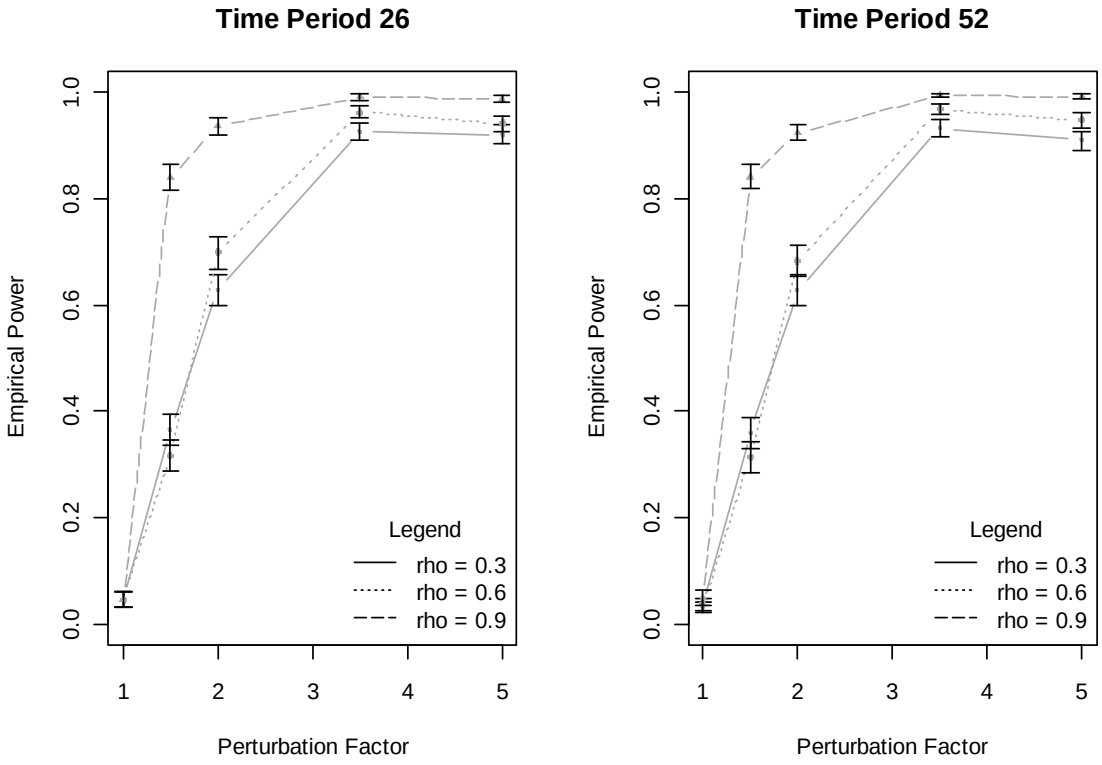


Figure 3-45: Plots of empirical power vs. perturbation factor for edge density = 0.9 and five edge perturbations; 3D configuration.



Inspection of figures 3-37 to 3-45 shows that for constant AR(1) correlation parameter, power increases as the perturbation factor increases. Two exception to this can be found in figures 3-38 and 3-45. In figures 3-38 and 3-45, the power for an AR(1) correlation of 0.3, the magnitude of power is higher for a perturbation factor of 3.5 versus the power for a perturbation factor of 5. However, the confidence intervals for power overlap for these cases, and thus are statistically tied.

3.2.2.3 Power versus AR(1) Correlation Parameter

Figure 3-46: Plots of empirical power vs. AR(1) correlation for edge density = 0.3 and a single edge perturbation; 3D configuration.

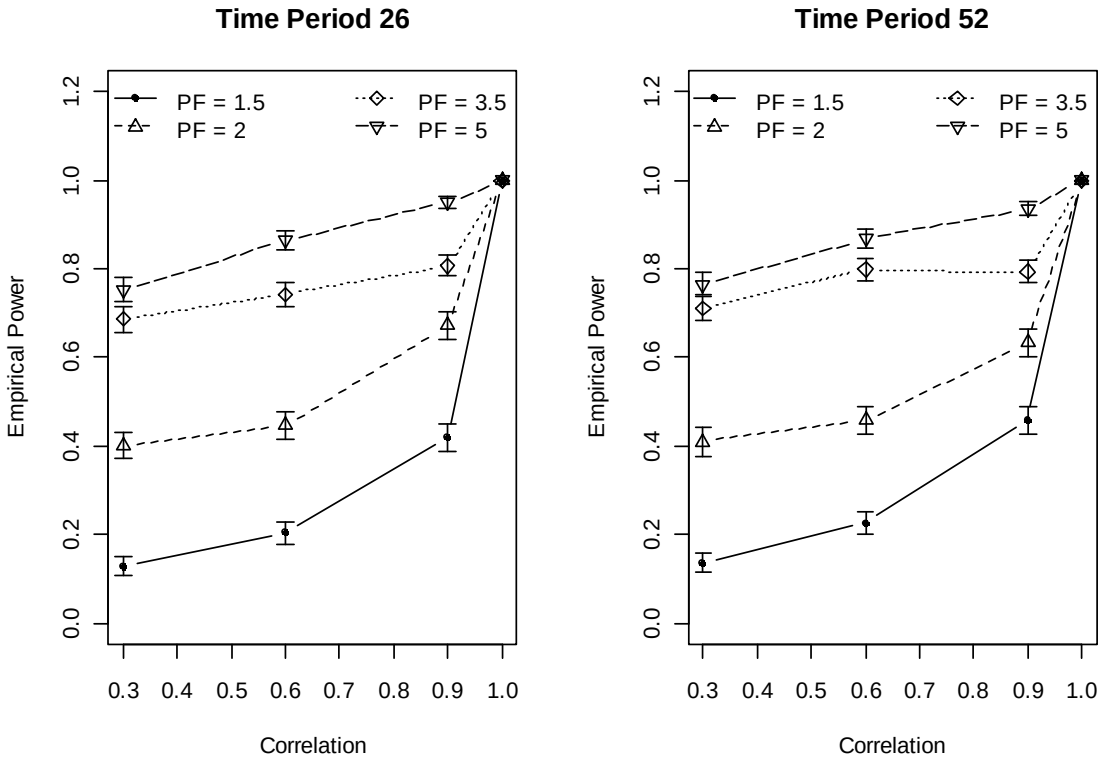


Figure 3-47: Plots of empirical power vs. AR(1) correlation for edge density = 0.3 and three edge perturbations; 3D configuration.

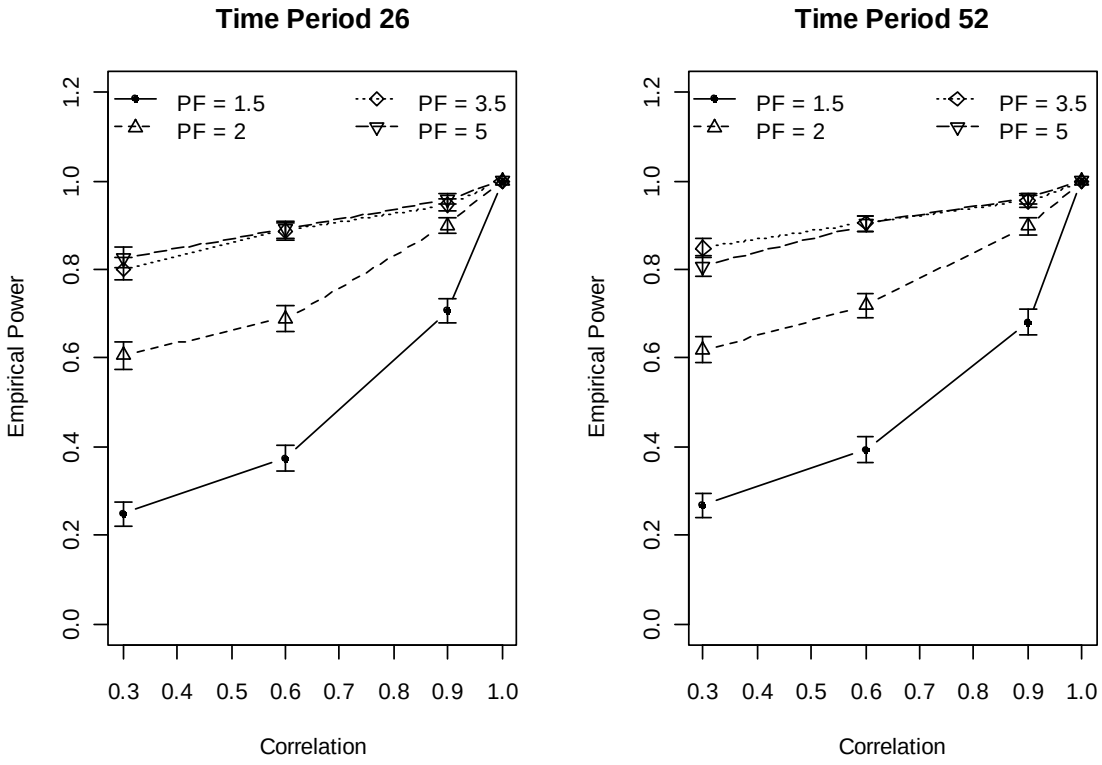


Figure 3-48: Plots of empirical power vs. AR(1) correlation for edge density = 0.3 and five edge perturbations; 3D configuration.

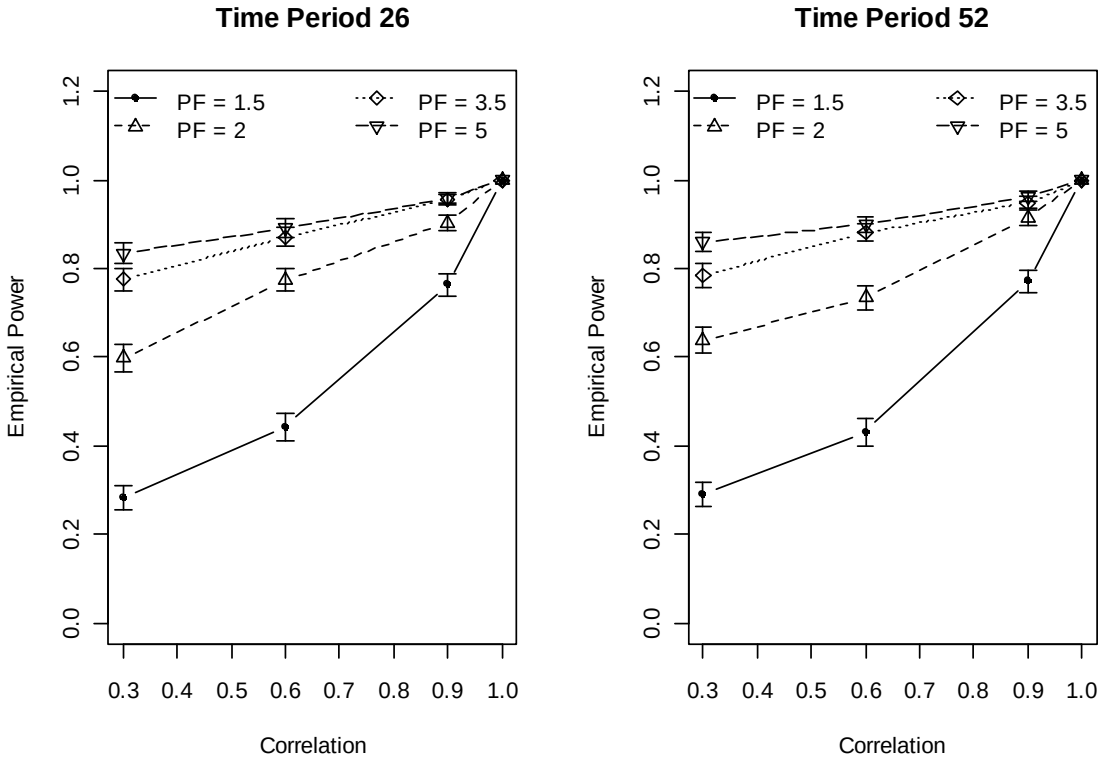


Figure 3-49: Plots of empirical power vs. AR(1) correlation for edge density = 0.6 and a single edge perturbation; 3D configuration.

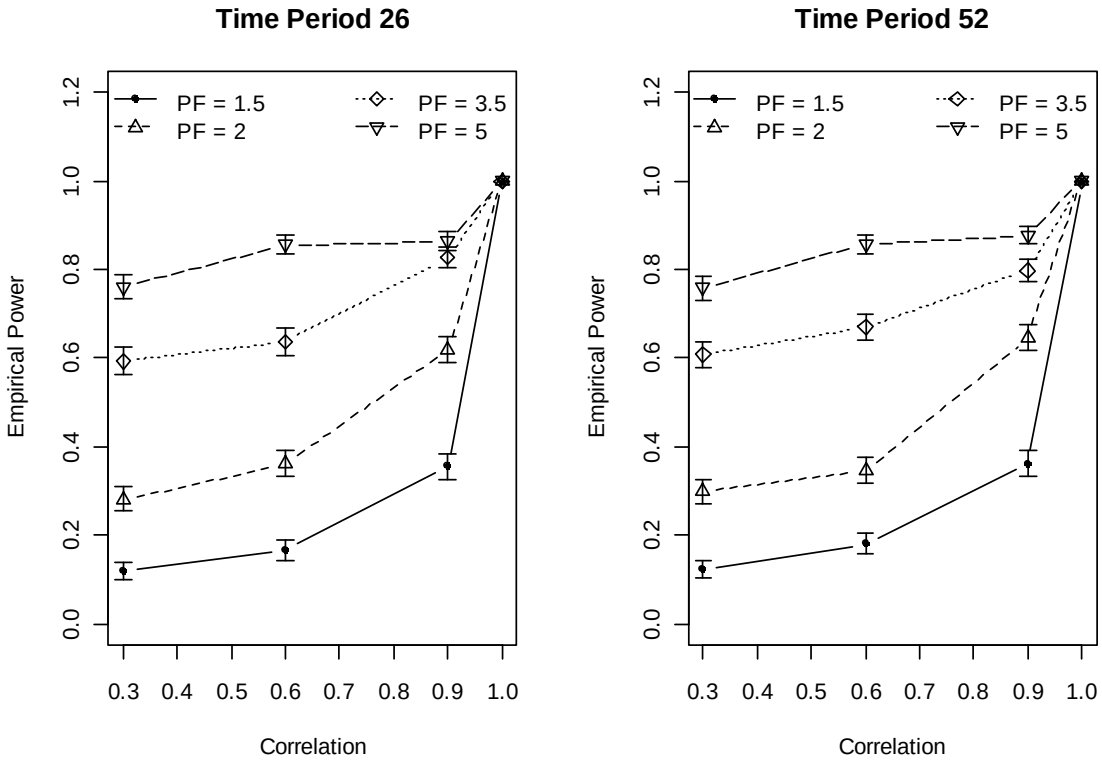


Figure 3-50: Plots of empirical power vs. AR(1) correlation for edge density = 0.6 and three edge perturbations; 3D configuration.

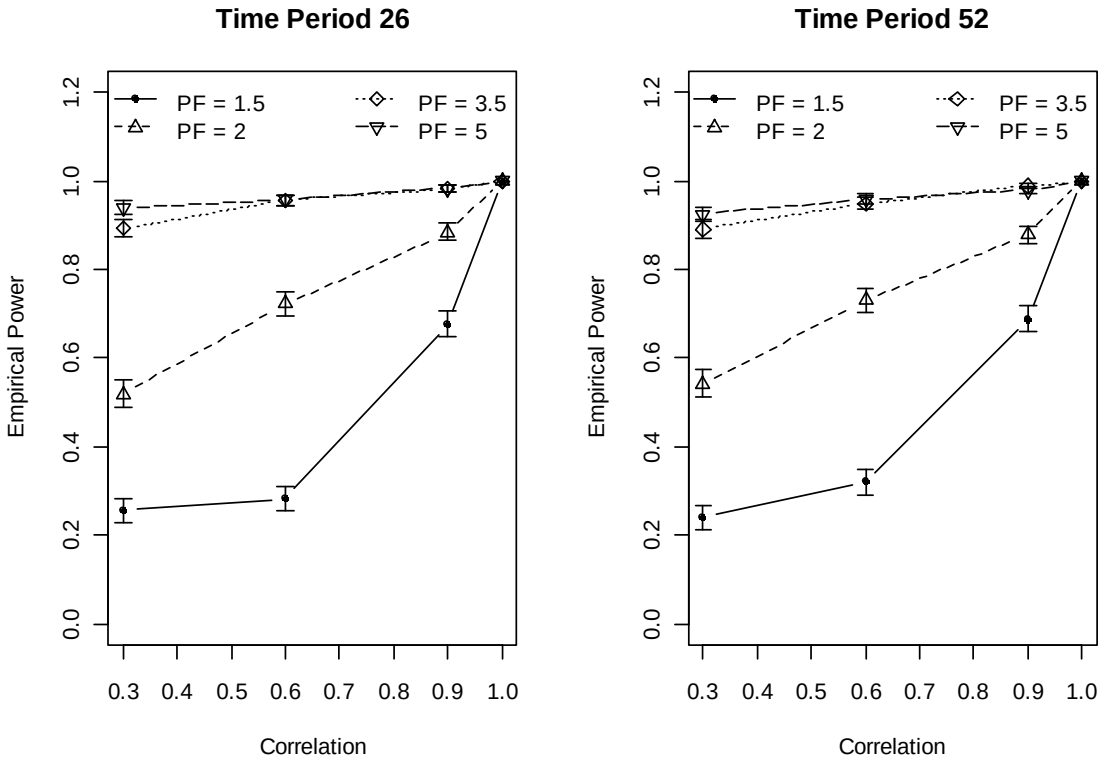


Figure 3-51: Plots of empirical power vs. AR(1) correlation for edge density = 0.6 and five edge perturbations; 3D configuration.

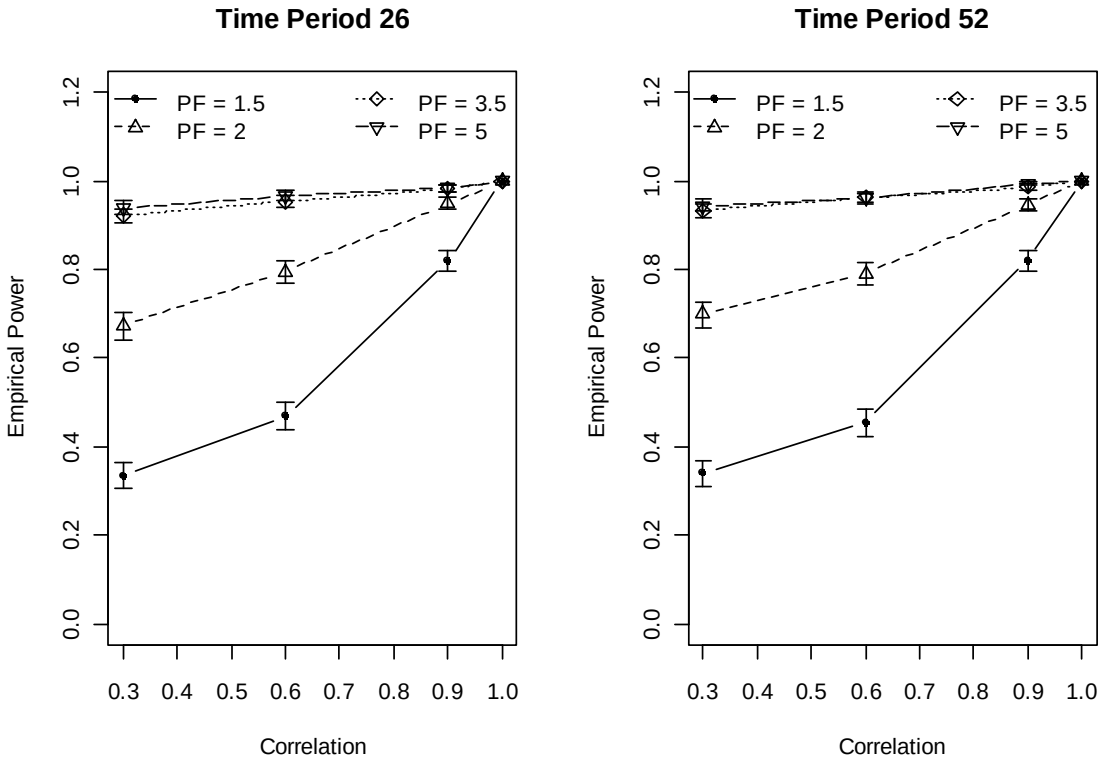


Figure 3-52: Plots of empirical power vs. AR(1) correlation for edge density = 0.9 and a single edge perturbation; 3D configuration.

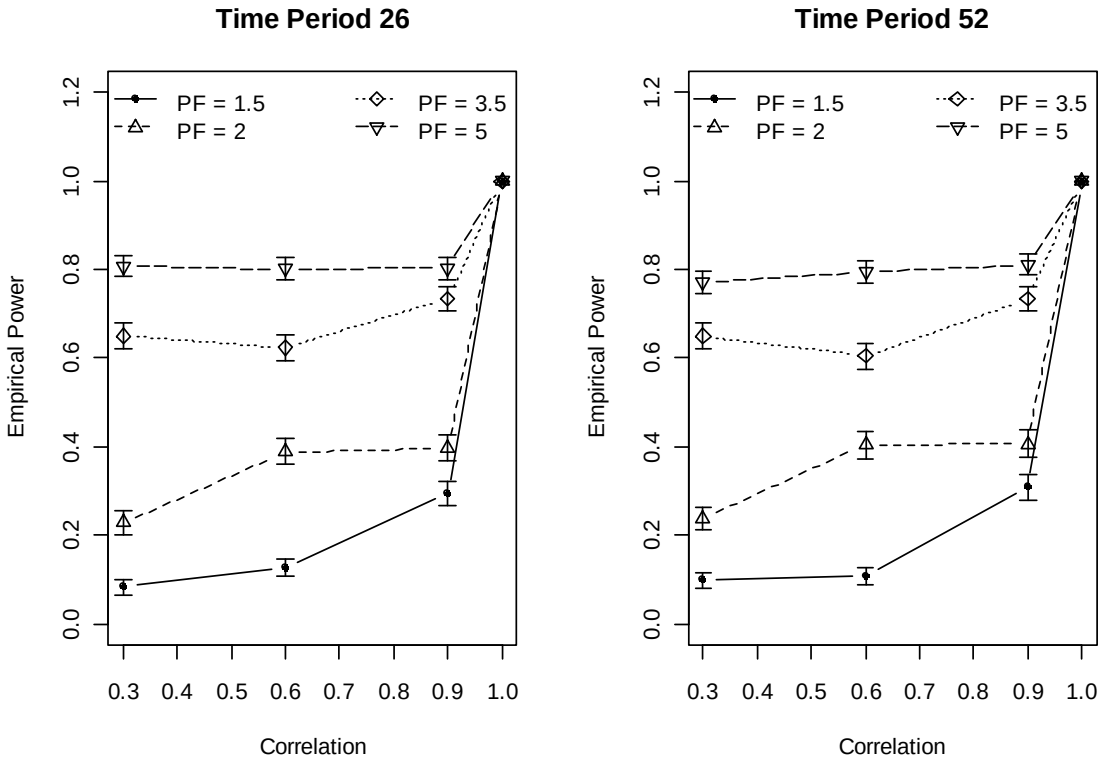


Figure 3-53: Plots of empirical power vs. AR(1) correlation for edge density = 0.9 and a three edge perturbation; 3D configuration.

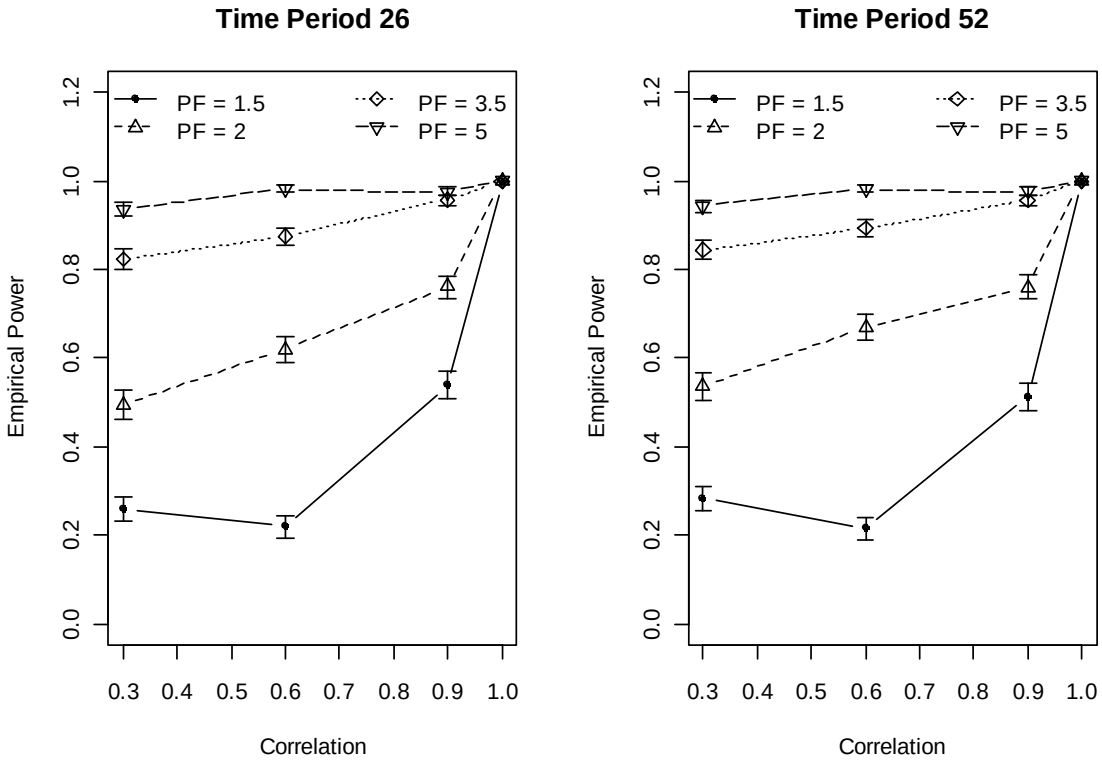
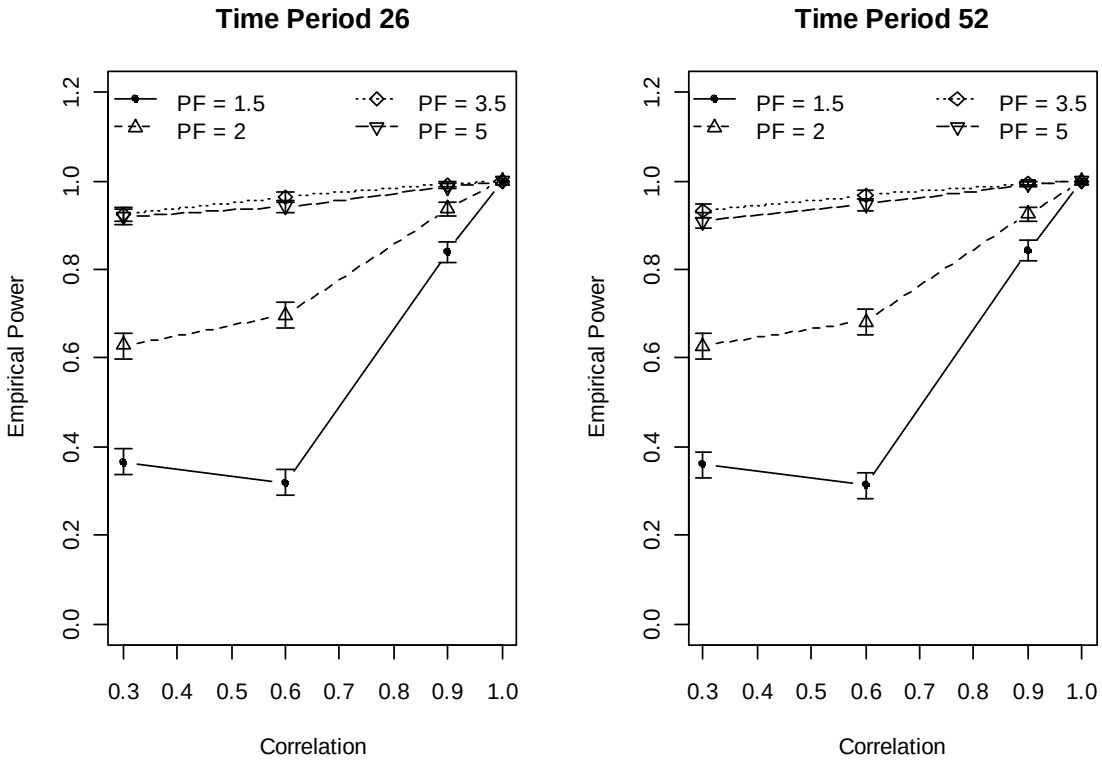


Figure 3-54: Plots of empirical power vs. AR(1) correlation for edge density = 0.9 and a five edge perturbation; 3D configuration.



Inspection of figures 3-46 to 3-54 reveals that power remains statistically constant (adjacent confidence intervals overlap) or increases as the AR(1) correlation parameter increases. The difference in power between lines of constant perturbation factor all decrease as the AR(1) correlation parameter increases, culminating in all combinations of factors yielding a power of 1 when the AR(1) correlation is 1.

3.2.3 Comparison of Power: Two- vs Three Dimensions

McNemar's test was used to determine if the dimension of the configuration significantly affects the empirical power of the test. This was done by analyzing the respective 2D and 3D vectors of 1's and 0's generated from the *power.calc* function using the base R function mcnemar.test. The p-values from McNemar's test can be found in tables 3-9 to 3-16. Of the 216 comparisons made (108 for each time of perturbation), 192 (98 for time = 26; 94 for time = 52) were significant at the 0.05 level. The difference in power was examined to determine which dimension was yielding the higher powers.

For perturbations occurring in the 52nd time period and an edge density of 0.9, there were four instances where power based on 2D configurations was higher than power based on 3D configurations. The largest difference was 0.119, and occurred for an AR(1) correlation of 0.3, and five edge perturbations at a perturbation factor of 1.5; the next largest difference was 0.097 and occurred for an AR(1) correlation of 0.6, and three edge perturbations at a perturbation factor of 1.5. The remaining seven differences were all less than 0.020. For edge densities of 0.3 and 0.6, each had two instances where power based on 2D configurations was higher than power based on 3D configurations; in neither case were the differences larger than 1%.

For perturbations occurring in the 26th time period and an edge density of 0.9, there were nine instances where power based on 2D configurations was higher than power based on 3D configurations. The largest difference was 0.179, and occurred for an

AR(1) correlation of 0.6, and five edge perturbations at a perturbation factor of 2; the next largest difference was 0.105 and occurred for an AR(1) correlation of 0.6, and three edge perturbations at a perturbation factor of 1.5. The remaining two differences were less than 0.021. For an edge density of 0.6, there were two instances where power based on 2D configurations was higher than power based on 3D configurations; in neither case was the difference larger than 1%. Based on these results, the preponderance of evidence suggests that taking into account the third dimension does increase the power of the method in detecting changes to the network.

Table 3-9: P-values from McNemar's test: comparing power for 2D versus 3D configurations; perturbations occurring in the 26th time period; perturbation factor = 1.5. An * indicates the difference in power favored 2D versus 3D.

No. Edges Perturbed	Edge Density	AR(1) Correlation Parameter		
		0.3	0.6	0.9
1	0.3	<0.0001*	<0.0001	<0.0001
	0.6	<0.0001	<0.0001*	<0.0001
	0.9	<0.0001*	<0.0001	<0.0001
3	0.3	<0.0001	<0.0001	0.0006
	0.6	<0.0001	<0.0001	<0.0001
	0.9	<0.0001	<0.0001*	<0.0001
5	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001*	<0.0001	0.8971
	0.9	<0.0001	<0.0001*	<0.0001

Table 3-10: P-values from McNemar's test: comparing power for 2D versus 3D configurations; perturbations occurring in the 26th time period; perturbation factor = 2. An * indicates the difference in power favored 2D versus 3D.

No. Edges Perturbed	Edge Density	AR(1) Correlation Parameter		
		0.3	0.6	0.9
1	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	<0.0001
	0.9	<0.0001	<0.0001	<0.0001*
3	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.3291*
	0.9	<0.0001*	<0.0001	<0.0001
5	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.0001
	0.9	<0.0001	0.6625*	<0.0001*

Table 3-11: P-values from McNemar's test: comparing power for 2D versus 3D configurations; perturbations occurring in the 26th time period; perturbation factor = 3.5. An * indicates the difference in power favored 2D versus 3D.

No. Edges Perturbed	Edge Density	AR(1) Correlation Parameter		
		0.3	0.6	0.9
1	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	<0.0001
	0.9	0.0048	0.7267*	<0.0001
3	0.3	<0.0001	<0.0001	0.0009
	0.6	<0.0001	<0.0001	0.0259
	0.9	0.0001*	0.2976*	0.1775*
5	0.3	<0.0001*	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.0003
	0.9	0.0450	<0.0001*	0.0442

Table 3-12: P-values from McNemar's test: comparing power for 2D versus 3D configurations; perturbations occurring in the 26th time period; perturbation factor = 5. An * indicates the difference in power favored 2D versus 3D.

No. Edges Perturbed	Edge Density	AR(1) Correlation Parameter		
		0.3	0.6	0.9
1	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.0256
	0.9	<0.0001	0.0001	<0.0001*
3	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	0.0725*	0.3074
	0.9	1.000	0.0053	<0.0001
5	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.0027
	0.9	<0.0001	0.1002*	<0.0001

Table 3-13: P-values from McNemar's test: comparing power for 2D versus 3D configurations; perturbations occurring in the 52nd time period; perturbation factor = 1.5. An * indicates the difference in power favored 2D versus 3D.

Edges Perturbed	Edge Density	Correlation		
		0.3	0.6	0.9
1	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001*	<0.0001
	0.9	<0.0001*	<0.0001	<0.0001
3	0.3	<0.0001	<0.0001	0.0042
	0.6	<0.0001	<0.0001	0.0012
	0.9	<0.0001	<0.0001*	<0.0001
5	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.1409
	0.9	<0.0001	0.0694*	<0.0001

Table 3-14: P-values from McNemar's test: comparing power for 2D versus 3D configurations; perturbations occurring in the 52nd time period; perturbation factor = 2. An * indicates the difference in power favored 2D versus 3D.

Edges Perturbed	Edge Density	Correlation		
		0.3	0.6	0.9
1	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	<0.0001
	0.9	<0.0001	<0.0001	<0.0001
3	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001*	0.9296*
	0.9	<0.0001	<0.0001	<0.0001
5	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	<0.0001
	0.9	<0.0001	0.0300*	0.0001*

Table 3-15: P-values from McNemar's test: comparing power for 2D versus 3D configurations; perturbations occurring in the 52nd time period; perturbation factor = 3.5. An * indicates the difference in power favored 2D versus 3D.

Edges Perturbed	Edge Density	Correlation		
		0.3	0.6	0.9
1	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	<0.0001
	0.9	0.0099	0.9240*	<0.0001
3	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.0003
	0.9	<0.0001	0.4094*	0.3711*
5	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.0001
	0.9	0.5827	<0.0001	0.3017

Table 3-16: P-values from McNemar's test: comparing power for 2D versus 3D configurations; perturbations occurring in the 52nd time period; perturbation factor = 5. An * indicates the difference in power favored 2D versus 3D.

Edges Perturbed	Edge Density	AR(1) Correlation Parameter		
		0.3	0.6	0.9
1	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.0450
	0.9	<0.0001	0.0509	<0.0001*
3	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001*	0.2864*	0.3105
	0.9	0.6892*	0.0218	<0.0001
5	0.3	<0.0001	<0.0001	<0.0001
	0.6	<0.0001	<0.0001	0.1213
	0.9	<0.0001	0.7003*	0.0433

3. 3 Conclusion

At the beginning of this chapter five questions were posed from which hypotheses were formulated. The first hypothesis was that for k out of m total non-zero edges (where $k < m$) the method will be able to detect a change in a closed network based solely on a change in the number of communications on the subset of k edges. This work showed that for the conditions assumed here, the method as proposed will detect changes in a closed network with reasonable power. With regard to the remaining hypotheses, it was shown that when considering only one factor changing at a time power increased as: the number of perturbed edges increased; the perturbation factor increased; the edge density increased; the AR(1) correlation parameter increased; and the dimension of the configuration increased.

3.4 References

Casella, G., and Berger, R. L. (2002). Statistical Inference. Duxbury Press.

Hunter, David R., Handcock, Mark S., Butts, Carter T., Goodreau, Steven M., Morris, Martina. (2008). ergm: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks. *Journal of Statistical Software*. Vol. 24, Issue 3.

McKenzie, E. (1988). Some ARMA Models for Dependent Sequences of Poisson Counts. *Advances in Applied Probability*. Vol. 20, No. 4, 822 – 835.

Milton, J. S. and Arnold, J. C. (1995). *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences, Third Edition*. McGraw Hill, Boston.

David Pollard. Chapter 13, Generating Functions and Transformations, STAT 241, Probability Theory with Applications Lecture Notes. Yale University, New Haven, CT. http://www.stat.yale.edu/~pollard/Courses/241.fall97/Generating.pdf

Snijders, Tom A.B., Pattison, Philippa E., Robins, Garry L., Handcock, Mark S. (2006) New Specifications for Exponential Random Graph Models. *Sociological Methodology*. Vol 36, 99-153.

Jürgen Symanzik. STAT 6710, Mathematical Statistics I Lecture Notes. Department of Mathematics and Statistics, Utah State University, Logan, UT. http://www.math.usu.edu/~symanzik/teaching/2008_stat6710/lect_main_full.pdf

Chapter 4

Limitations and Future Work

As the reader is undoubtedly aware, the method presented in this study has some limitations. First, a Poisson AR(1) process was used to simulate the number of communications between nodes. Obviously, there is no guarantee that a real world closed network will have a structured correlation, let alone one with an AR(1) structure. The robustness of the method must be tested via simulation using various correlation structures, as well as ultimately being tested using real-world closed networks.

The crux of the method relies on a history of "normal" communications being catalogued, and the sequential aR^2's being recorded. Practically speaking, a minimum history of 20 "normal" time periods should be collected before testing future time periods ($t \geq 21$) for a change from "normal". The 20 time period minimum is suggested assuming a significance level of 0.05 will be used to detect a change (i.e. 20*0.05 = 1). From an implementation stand point, the collection of the network history under "normal" conditions may be the most glaring limitation of the method. First, one must already have a notion as to what constitutes "normal". Second, even if the starting point is at a "normal" stage in communications, there is no guarantee that the first time period will elapse without an "abnormality" occurring.

The method also relies on configuration matching techniques that can "account" for scaling. As such, if all edges in a closed network see a near identical increase in the number of communications, the method as proposed will likely not detect the change. If

by chance, all nodes see exactly the same multiplicative change or no change at all, then the aR^2 value between these two time periods will be one. However, for near equal perturbation factors on all non-zero edges, it may be possible to detect "abnormally" high aR^2 values. This investigation is left for future work.

Another question that was not addressed is: What happens if there is both a structural (addition or deletion of an edge) as well as a change in the number of communications between nodes? In real world networks, this is probably a likely occurrence. The method as presented has no way of accounting for this eventuality. However, in practice, the user of this method should find the graph correlation as outlined in chapter one prior to implementing the method described in this dissertation. If the graph correlation is 1, and a change is detected, then the change is due to a change in the number of communications on at least one edge[8]. If the graph correlation is less than 1, then any change due to a shift in the number of communications between nodes will be confounded with the structural change in the network. Once more, this is an area for future study.

A limitation of the simulation study undertaken is that only a single replicate of power was obtained for each edge density, AR(1) correlation parameter, number of edges perturbed, perturbation factor, and time period of change combination. To truly understand the behavior of power, a broader simulation study needs to be undertaken in which each simulation presented here is replicated a sufficient number of times (i.e.

[8] This assumes that the network in question is closed, and that the communications pattern is governed by an AR(1) Poisson process.

1000) to obtain the sampling distribution for empirical power at each combination of factors; this is left for future work.

In addition, the effect of network size on the ability to detect a change needs to be explored. As this method was conceived for closed networks, the reasonable network sizes to explore are 10 to 50 in increments of 5. It should be noted that as the network size increases, the minimum edge density of interest will decrease. For example, a network of size 30 has a total of 435 possible edges (symmetric network); the minimum number of edges to obtain a linear or circular chain is 29 and 30, respectively. This yields a potentially trivial edge density of 0.06; adding either 5% of 10% of the total possible edges to 30 gives minimum edge densities of 0.12 and 0.17, respectively. Additionally, it is reasonable to expect that as network size increases, the size of perturbation that can be detected with decent power will also increase. The size of perturbation has two components, number of edges perturbed and perturbation factor; each needs to be explored individually as well as in combination. Finally, the minimum configuration dimension that yields the most consistent power as a function of network size needs to be explored. Recall in section 2.5, that there are instances when the aR^2 value levels or drops slightly prior to resuming an increasing value; the dimension just prior to the first leveling or dip may provide insight into the minimum number of dimensions for capturing the configuration of the network. These investigations are left for future work.

Appendix A

R functions and simulation code written to support the work in this dissertation.

The function *edge.sim* generates adjacency matrices based on the number of nodes, *n*, and the desired edge density, *p*.

```
edge.sim <- function(n, p){
  edge.out <- matrix(0, nrow = n, ncol = n)
    b <- (n - 1)
    for (i in 1:b){
      a <- (i + 1)
      for (j in a:n){
        edge.out[i, j] <- rbinom(1, 1, p)
        }
      }
```

The function *lambda* assigns a mean number of communications to each edge. Inputs to the function are n, the network size; a and b, the alpha and beta parameters for the gamma distribution. Note, in the R language rgamma function is parameterized such that the mean is a/b.

```
lambda <- function(n, a, b){
  mus.out <- matrix(0, nrow = n, ncol = n)
      b <- (n - 1)
    for (i in 1:b){
      a <- (i + 1)
      for (j in a:n){
        mus.out[i, j] <- round(rgamma(1, a, 1/b), 0)
        }
      }
  mus.out
  }
```

The function *mu.0* randomly generates the first observed number of communications between any two nodes that have a non-zero edge. Inputs to the function are: n, the network size; rho, the AR(1) correlation parameter; d, the adjacency matrix; and l, the output of the function *lambda*.

```
mu.0 <- function(n, rho, d, l){
  inter.mu <- matrix(0, nrow = n, ncol = n)
    b <- n-1
    for (i in 1:b){
       a <- (i + 1)
    for (j in a:n){
       mu.3 <- rho*l[i,j]
      inter.mu[i,j] <- ifelse(d[i,j]==1, rpois(1, mu.3), 0)
    }
  }
inter.mu }
```

The function AR.wgt yields edge weights with an ar1 correlation structure based on Mckenzie's (1988) Poisson AR(1) process. Here, n = the number of nodes in the network; rho = correlation / prob of success in the binomial component; d = the adjacency (10x10) matrix of 1's and 0's; l = the matrix of means for each node to node intersection; and x.0 = the initial random poisson mean value for each node to node communication edge.

```
AR.wgt <- function(n, rho, d, l, x.0){
ar.dis <- matrix(0, nrow = n, ncol = n)
b <- n-1
for (i in 1:b){
  a <- (i + 1)
  for (j in a:n){
     nn <- x.0[i,j]
     ar.dis[i,j] <- ifelse(d[i,j]==1, rbinom(1, nn, rho) + rpois(1,
l[i,j]*(1-rho)), 0)
     }
  }
ar.dis
}
```

The functions, *one.perts, two.perts, ..., five.perts* are used to randomly perturb 1, 2, …, 5 edges, respectfully. Inputs to the function are the sub weighted adjacency matrix associated with a given time period in which the change is to occur, and the size of the perturbation. The size must be greater than 0.

```
one.perts <- function(TP, size){
N <- ncol(TP)
Rep <- nrow(TP)/N
col.0 <- 0
for (j in 1:Rep){
  UP   <- j*N
  DOWN <- UP - (N-1)
  D.0 <- DOWN - 1
  inter.vec <- vec(TP[DOWN:UP, ])
  non.0.pos <- not.0(inter.vec)
  pos <- sample(non.0.pos, 1)
  a.1 <- inter.vec[pos[1], 2]
  b.1 <- inter.vec[pos[1], 3]
    TP[(D.0 + a.1), (col.0 + b.1)] <- size*TP[(D.0 + a.1), (col.0 +
b.1)]
  }
  TP
}


two.perts <- function(TP, size){
N <- ncol(TP)
Rep <- nrow(TP)/N
col.0 <- 0
for (j in 1:Rep){
  UP   <- j*N
```

```
  DOWN <- UP - (N-1)
  D.0 <- DOWN - 1
  inter.vec <- vec(TP[DOWN:UP, ])
  non.0.pos <- not.0(inter.vec)
  pos <- sample(non.0.pos, 2)
  a.1 <- inter.vec[pos[1], 2]
  b.1 <- inter.vec[pos[1], 3]
  a.2 <- inter.vec[pos[2], 2]
  b.2 <- inter.vec[pos[2], 3]
    TP[(D.0 + a.1), (col.0 + b.1)] <- size*TP[(D.0 + a.1), (col.0 +
b.1)]
    TP[(D.0 + a.2), (col.0 + b.2)] <- size*TP[(D.0 + a.2), (col.0 +
b.2)]
  }
  TP
}

three.perts <- function(TP, size){
N <- ncol(TP)
Rep <- nrow(TP)/N
col.0 <- 0
for (j in 1:Rep){
  UP   <- j*N
  DOWN <- UP - (N-1)
  D.0 <- DOWN - 1
  inter.vec <- vec(TP[DOWN:UP, ])
  non.0.pos <- not.0(inter.vec)
  pos <- sample(non.0.pos, 3)
  a.1 <- inter.vec[pos[1], 2]
  b.1 <- inter.vec[pos[1], 3]
  a.2 <- inter.vec[pos[2], 2]
  b.2 <- inter.vec[pos[2], 3]
  a.3 <- inter.vec[pos[3], 2]
  b.3 <- inter.vec[pos[3], 3]
    TP[(D.0 + a.1), (col.0 + b.1)] <- size*TP[(D.0 + a.1), (col.0 +
b.1)]
    TP[(D.0 + a.2), (col.0 + b.2)] <- size*TP[(D.0 + a.2), (col.0 +
b.2)]
    TP[(D.0 + a.3), (col.0 + b.3)] <- size*TP[(D.0 + a.3), (col.0 +
b.3)]
  }
  TP
}

four.perts <- function(TP, size){
N <- ncol(TP)
Rep <- nrow(TP)/N
col.0 <- 0
for (j in 1:Rep){
  UP   <- j*N
  DOWN <- UP - (N-1)
  D.0 <- DOWN - 1
  inter.vec <- vec(TP[DOWN:UP, ])
  non.0.pos <- not.0(inter.vec)
```

```
  pos <- sample(non.0.pos, 4)
  a.1 <- inter.vec[pos[1], 2]
  b.1 <- inter.vec[pos[1], 3]
  a.2 <- inter.vec[pos[2], 2]
  b.2 <- inter.vec[pos[2], 3]
  a.3 <- inter.vec[pos[3], 2]
  b.3 <- inter.vec[pos[3], 3]
  a.4 <- inter.vec[pos[4], 2]
  b.4 <- inter.vec[pos[4], 3]
    TP[(D.0 + a.1), (col.0 + b.1)] <- size*TP[(D.0 + a.1), (col.0 +
b.1)]
    TP[(D.0 + a.2), (col.0 + b.2)] <- size*TP[(D.0 + a.2), (col.0 +
b.2)]
    TP[(D.0 + a.3), (col.0 + b.3)] <- size*TP[(D.0 + a.3), (col.0 +
b.3)]
    TP[(D.0 + a.4), (col.0 + b.4)] <- size*TP[(D.0 + a.4), (col.0 +
b.4)]
  }
  TP
}

five.perts <- function(TP, size){
N <- ncol(TP)
Rep <- nrow(TP)/N
col.0 <- 0
for (j in 1:Rep){
  UP   <- j*N
  DOWN <- UP - (N-1)
  D.0 <- DOWN - 1
  inter.vec <- vec(TP[DOWN:UP, ])
  non.0.pos <- not.0(inter.vec)
  pos <- sample(non.0.pos, 5)
  a.1 <- inter.vec[pos[1], 2]
  b.1 <- inter.vec[pos[1], 3]
  a.2 <- inter.vec[pos[2], 2]
  b.2 <- inter.vec[pos[2], 3]
  a.3 <- inter.vec[pos[3], 2]
  b.3 <- inter.vec[pos[3], 3]
  a.4 <- inter.vec[pos[4], 2]
  b.4 <- inter.vec[pos[4], 3]
  a.5 <- inter.vec[pos[5], 2]
  b.5 <- inter.vec[pos[5], 3]
    TP[(D.0 + a.1), (col.0 + b.1)] <- size*TP[(D.0 + a.1), (col.0 +
b.1)]
    TP[(D.0 + a.2), (col.0 + b.2)] <- size*TP[(D.0 + a.2), (col.0 +
b.2)]
    TP[(D.0 + a.3), (col.0 + b.3)] <- size*TP[(D.0 + a.3), (col.0 +
b.3)]
    TP[(D.0 + a.4), (col.0 + b.4)] <- size*TP[(D.0 + a.4), (col.0 +
b.4)]
    TP[(D.0 + a.5), (col.0 + b.5)] <- size*TP[(D.0 + a.5), (col.0 +
b.5)]
  }
  TP}
```

The function *wn.mds* implements Torgerson's metric multidimensional scaling algorithm after ensuring that the network under consideration is symmetric. Inputs to the function are X, the upper (lower) triangular weighted adjacency matrix and d, the desired number of dimensions for the configuration.

```
wn.mds <- function(X, d){
  Y <- X + t(X)
  out <- cmdscale(Y, d)
  out}
```

There are no functions or packages in R that calculate the trace of a matrix; to ease coding the function *tr* was written to fill this gap.

```
tr <- function(m){sum(diag(m))}
```

The function *procrustes* implements the procrustes procedure as described in chapter 1. The Cholesky root (R function chol) is used to calculate the square root of a matrix. The only inputs to this function are the two configurations to be matched. As written, this function rotates and scales the first matrix entered to match the configuration of the second matrix.

```
procrustes <- function(x, y){
c.x <- x - matrix(colMeans(x), nrow = nrow(x), ncol = ncol(x))
c.y <- y - matrix(colMeans(y), nrow = nrow(y), ncol = ncol(y))
n  <- nrow(c.x)
ypx <- t(c.y)%*%c.x # E = Y'X
xpyypx <- t(ypx)%*%ypx
A <- chol(xpyypx)%*%solve(ypx) # Find Rotation Matrix
X <- c.x%*%A
rho <- tr(chol(xpyypx))/tr(t(c.x)%*%c.x)
X.scale <- rho*X
ypx.rot <- t(c.y)%*%X.scale
R.2 <-
((tr(chol(t(ypx.rot)%*%ypx.rot)))^2)*((tr(t(X.scale)%*%X.scale)*tr(t(c.
y)%*%c.y))^-1)
}
```

The *euclidean* function implements Tobler's Euclidean transformation. The inputs to this function are the two configurations to be matched; and the first configuration is matched to the second configuration entered.

```
euclidean <- function(z, w){
n <- nrow(z)
A <- matrix(c( n, 0, sum(z[,1]), -sum(z[,2]), 0, n, sum(z[,2]),
      sum(z[,1]),sum(z[,1]), sum(z[,2]), sum(z[,1]^2 + z[,2]^2),  0,
     -sum(z[,2]), sum(z[,1]), 0,  sum(z[,2]^2 + z[,1]^2)),
      nrow = 4, ncol = 4, byrow = T)

B <- matrix(c(sum(w[,1]), sum(w[,2]), sum(w[,1]*z[,1] + w[,2]*z[,2]),
              sum(w[,2]*z[,1] - w[,1]*z[,2] )), nrow = 4, ncol = 1)
X <- solve(A)%*%B
W.hat = matrix(c(rep(X[1], n), rep(X[2], n)), nrow = n, ncol = 2) +
```

```
        matrix(c((X[3]*z[,1] - X[4]*z[,2]),(X[3]*z[,2] + X[4]*z[,1])),
nrow = n,ncol = 2)
e <- w - W.hat
M <- matrix(c(rep(mean(w[,1]), n), rep(mean(w[,2]), n)), nrow = n, ncol
= 2)
W.cen <- w - M
r2 <- (sum(diag(W.cen%*%t(W.cen))) - sum(diag(e%*%t(e))))
/sum(diag(W.cen%*%t(W.cen)));

r2
  }
```

The *gen.affine* function implements Tobler's affine transformation. The inputs to this function are the two configurations to be matched; the first configuration is matched to the second configuration entered.

```
gen.affine <- function(z, w){
  n <- nrow(z)
  X <- matrix(c(rep(1, n), z), nrow = n, ncol = (ncol(z)+1))
  XPX <- t(X)%*%X
  XPY <- t(X)%*%w
  B <- ginv(XPX)%*%XPY
  W.hat <- X%*%B
  e <- w - W.hat
  M <- matrix(colMeans(w), nrow = n, ncol = ncol(w), byrow = T)
  W.cen <- w - M
  r2 <- (sum(diag(W.cen%*%t(W.cen))) - sum(diag(e%*%t(e))))
/sum(diag(W.cen%*%t(W.cen)));
  #list('R.sq' = r2, 'Beta' = B, 'Dimension' = ncol(w))
  r2 }
```

The function *power.calc* is used to calculate the empirical power for changes that occur at two different time periods simultaneously. Inputs to this function are: A, the matrix of aR^2's for the "normal" network conditions; B, the matrix of aR^2 for the first perturbed time period; E, the matrix of aR^2 for the second perturbed time period; and P and Rho, the edge density and AR(1) correlation parameter, respectively, of the networks from which the aR2's were found. These last two inputs are only required to help the user identify for which combination of network factors the power was calculated for; they have no effect on the power calculation.

```
pwr.calc <- function(A, B, E, P, Rho){
Rep2 <- nrow(A)
his.26 <- as.matrix(A[,2:24]) # Drop the first tp b/c it is a seed
value
his.52 <- as.matrix(A[,2:50]) # Drop the first tp b/c it is a seed
value
B <- as.matrix(B)
E <- as.matrix(E)
pwr.mat.26a <- matrix(NA, nrow = Rep2, ncol = 1)
pwr.mat.52a <- matrix(NA, nrow = Rep2, ncol = 1)
for (j in 1:Rep2){
  sort.26 <- sort(his.26[j,])
```

```
  sort.52 <- sort(his.52[j,])
  CV.26 <- sort.26[1]
  CV.52 <- sort.52[2]
  pwr.mat.26a[j] <- ifelse(B[j] < CV.26, 1, 0)
  pwr.mat.52a[j] <- ifelse(E[j] < CV.52, 1, 0)
}
pwr.mat.26a <- pwr.mat.26a[complete.cases(pwr.mat.26a)]
a1 <- length(pwr.mat.26a)
pwr.mat.52a <- pwr.mat.52a[complete.cases(pwr.mat.52a)]
a2 <- length(pwr.mat.52a)
pwr.26a <- sum(pwr.mat.26a)/a1
pwr.52a <- sum(pwr.mat.52a)/a2
pwr <- matrix(c(P, Rho,  round(pwr.26a,3), round(pwr.52a,3)),
       nrow = 1, ncol = 4)
list('Power' = pwr, 'tp26' = pwr.mat.26a, 'tp52' = pwr.mat.52a)
}

N <- 10  #number of nodes
Rep <- 1000 # number of replications
rep.2 <- 100
tp <- 52 # number of time periods to simulate
a.1 <- 4
b.1 <- 1/17
set.seed(29); P = 0.9; RHO = 0.3
L  <- lambda(N, a.1, b.1) # generates the matrix of Poisson means

P9R3 <- matrix(NA, nrow = N*Rep, ncol = tp*N)
edges <- matrix(NA, nrow = N*Rep, ncol = tp*N)
for (j in 1:Rep){
 UP   <- j*N
 DOWN <- UP - (N-1)
 E1 <- edge.sim2(N, P)
 P9R3[DOWN:UP, 1:N] <- mu.0(N, RHO, E1, L)
for (i in 2:tp){
 up <- i*N
 low <- up - (N-1)
 lagup <- up - N
 laglow <- low-N
 edges[DOWN:UP,low:up] <- E1
 P9R3[DOWN:UP,low:up] <- AR.wgt(N, RHO, E1, L, P9R3[DOWN:UP,laglow:lagup])
 }
 }
# Perturb a time period.  Here three randomly chosen edges are being
# perturbed by a factor of 3.5 in the 26th and 52nd time periods
P9R3.26 <- P9R3[,251:260]
P9R3.52 <- P9R3[,511:520]
amt <- 3.5
set.seed(4671)
P9R3.26.pert <-  three.perts(P9R3.26, amt)
P9R3.52.pert <-  three.perts(P9R3.52, amt)
```

```
#Get the 2D and 3D configurations for each network / time period
P9R3.coords <- matrix(NA, nrow = nrow(P9R3), ncol = tp*3)
for (j in 1:Rep){
  dd <- N*j
  cc <- dd - N + 1
for (i in 1:tp){
  b <- N*i
  a <- b - N + 1
  bb <- 3*i
  aa <- bb - 2
  P9R3.coords[cc:dd,aa:bb]  <- wn.mds(P9R3[cc:dd,a:b], 3)
}
}
d2 <- seq(3, 156, 3)
P9R3.coords.2D <- P9R3.coords[, -d2] # 2D coords
P9R3.coords.3D <- P9R3.coords        # 3D coords

# Find the sequential aR2's

P9R3.2d.Rsq <- matrix(NA, nrow = Rep, ncol = 51)
P9R3.3d.Rsq <- matrix(NA, nrow = Rep, ncol = 51)
for (j in 1:Rep){
  UP   <- j*N
  DOWN <- UP - (N-1)
for (i in 1:51){
  b <- 3*(i)
  a <- b-2
  bb <- b + 3
  aa <- a + 3
  g <- 2*i
  f <- g - 1
  gg <- g + 2
  ff <- f + 2
  P9R3.2d.Rsq[j,i] <- gen.affine(P9R3.coords.2D[DOWN:UP,f:g], P9R3.coords.2D[DOWN:UP,
ff:gg])
  P9R3.3d.Rsq[j,i] <- gen.affine(P9R3.coords.3D[DOWN:UP,a:b], P9R3.coords.3D[DOWN:UP,
aa:bb])
  }
}

# Get the coords for the perturbed time periods
P9R3.26.coords <- matrix(NA, nrow = 10000, ncol = 3)
P9R3.52.coords <- matrix(NA, nrow = 10000, ncol = 3)
for (j in 1:Rep){
  UP   <- j*10
  DOWN <- UP - 9
  P9R3.26.coords[DOWN:UP,] <- wn.mds(P9R3.26.pert[DOWN:UP,], 3)
  P9R3.52.coords[DOWN:UP,] <- wn.mds(P9R3.52.pert[DOWN:UP,], 3)
```

```
  }
P9R3.26.2d <- P9R3.26.coords[, 1:2]
P9R3.52.2d <- P9R3.52.coords[, 1:2]
P9R3.26.3d <- P9R3.26.coords
P9R3.52.3d <- P9R3.52.coords

# Find Sequential aR2's for the two perturbed time periods
P9R3.25.2d <- P9R3.coords.2D[, 49:50]
P9R3.51.2d <- P9R3.coords.2D[, 103:104]
P9R3.25.3d <- P9R3.coords.3D[, 73:75]
P9R3.51.3d <- P9R3.coords.3D[, 151:153]
P9R3.2526.2d <-   matrix(NA, nrow = 1000, ncol = 1)
P9R3.5152.2d <-   matrix(NA, nrow = 1000, ncol = 1)
P9R3.2526.3d <-   matrix(NA, nrow = 1000, ncol = 1)
P9R3.5152.3d <-   matrix(NA, nrow = 1000, ncol = 1)

for (i in 1:1000){
  b <- 10*i
  a <- b - 9
  P9R3.2526.2d[i] <-  gen.affine(P9R3.25.2d[a:b,], P9R3.26.2d[a:b,])
  P9R3.5152.2d[i] <-  gen.affine(P9R3.51.2d[a:b,], P9R3.52.2d[a:b,])
  P9R3.2526.3d[i] <-  gen.affine(P9R3.25.3d[a:b,], P9R3.26.3d[a:b,])
  P9R3.5152.3d[i] <-  gen.affine(P9R3.51.3d[a:b,], P9R3.52.3d[a:b,])
  }
# Calculate the Empirical Power

P.93.2d <- pwr.calc(P9R3.2d.Rsq, P9R3.2526.2d, P9R3.5152.2d, 0.9, 0.3)
P.93.3d <- pwr.calc(P9R3.3d.Rsq, P9R3.2526.3d, P9R3.5152.3d, 0.9, 0.3)
Pwr.93.2d <- P.93.2d$Power
Pwr.93.3d <- P.93.3d$Power
colnames(Pwr.93.2d) <- c('Density', 'Corr', 'Power.26', 'Power.52')
colnames(Pwr.93.3d) <- c('Density', 'Corr', 'Power.26', 'Power.52')
Pwr.93.2d  # Prints the power for 2D configuration to the screen
Pwr.93.3d  # Prints the power for 3D configuration to the screen
```